



**NAVAL  
POSTGRADUATE  
SCHOOL**

**MONTEREY, CALIFORNIA**

**THESIS**

**EFFICIENT IMPLEMENTATION OF FILTERING AND  
RESAMPLING OPERATIONS ON FIELD  
PROGRAMMABLE GATE ARRAYS (FPGAs) FOR  
SOFTWARE DEFINED RADIO (SDR)**

by

Georgios Giannoulis

December 2008

Thesis Advisor:  
Thesis Co-advisor:

Roberto Cristi  
Craig W. Rasmussen

**Approved for public release; distribution is unlimited**

THIS PAGE INTENTIONALLY LEFT BLANK

<b>REPORT DOCUMENTATION PAGE</b>			<i>Form Approved OMB No. 0704-0188</i>
Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instruction, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188) Washington DC 20503.			
<b>1. AGENCY USE ONLY (Leave blank)</b>	<b>2. REPORT DATE</b> December 2008	<b>3. REPORT TYPE AND DATES COVERED</b> Master's Thesis	
<b>4. TITLE AND SUBTITLE</b> Efficient Implementation of Filtering and Resampling Operations on Field Programmable Gate Arrays (FPGAs) for Software Defined Radio (SDR).		<b>5. FUNDING NUMBERS</b>	
<b>6. AUTHOR(S)</b> Georgios Giannoulis		<b>8. PERFORMING ORGANIZATION REPORT NUMBER</b>	
<b>7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)</b> Naval Postgraduate School Monterey, CA 93943-5000		<b>10. SPONSORING/MONITORING AGENCY REPORT NUMBER</b>	
<b>9. SPONSORING /MONITORING AGENCY NAME(S) AND ADDRESS(ES)</b> N/A		<b>11. SUPPLEMENTARY NOTES</b> The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.	
<b>12a. DISTRIBUTION / AVAILABILITY STATEMENT</b> Approved for public release; distribution is unlimited		<b>12b. DISTRIBUTION CODE</b>	
<b>13. ABSTRACT (maximum 200 words)</b> In Software Defined Radios a good portion (or even the entirety) of the modulation and demodulation processes is performed in the digital domain. The data rate of the transmitted information is very important, since efficiency is a key requirement in real time implementations and cost increases considerably with the number of samples per second to be processed. In this thesis, we address the problem of efficient design of the resampling operations, so that they can be implemented on Field Programmable Gate Arrays (FPGAs). A set of filtering and resampling operations is developed in the Simulink environment through Xilinx/Simulink blocksets, where all the included subsystems of the design are fully accessible by the designer in any stage of operation. The key ingredient is the use of a Multiplier and Accumulator (MAC) architecture, which can be either time multiplexed for maximum hardware efficiency, or run on a parallel structure for maximum time efficiency.			
<b>14. SUBJECT TERMS</b> Digital Signal Processing, Software Defined Radio, Field Programmable Gate Array, Resampling, Filtering, Xilinx, System Generator.		<b>15. NUMBER OF PAGES</b> 79	
		<b>16. PRICE CODE</b>	
<b>17. SECURITY CLASSIFICATION OF REPORT</b> Unclassified	<b>18. SECURITY CLASSIFICATION OF THIS PAGE</b> Unclassified	<b>19. SECURITY CLASSIFICATION OF ABSTRACT</b> Unclassified	<b>20. LIMITATION OF ABSTRACT</b> UU

NSN 7540-01-280-5500

Standard Form 298 (Rev. 8-98)  
Prescribed by ANSI Std. Z39.18

THIS PAGE INTENTIONALLY LEFT BLANK

**Approved for public release; distribution is unlimited**

**EFFICIENT IMPLEMENTATION OF FILTERING AND RESAMPLING  
OPERATIONS ON FIELD PROGRAMMABLE GATE ARRAYS (FPGAs) FOR  
SOFTWARE DEFINED RADIO (SDR)**

Georgios Giannoulis  
Lieutenant, Hellenic Navy  
B.S., Hellenic Naval Academy, 1997

Submitted in partial fulfillment of the  
requirements for the degrees of

**MASTER OF SCIENCE IN ELECTRICAL ENGINEERING  
and  
MASTER OF SCIENCE IN APPLIED MATHEMATICS**

from the

**NAVAL POSTGRADUATE SCHOOL  
December 2008**

Author: Georgios Giannoulis

Approved by: Roberto Cristi  
Thesis Advisor

Craig W. Rasmussen  
Thesis Co-Advisor

Jeffrey Knorr  
Chairman, Department of Electrical and Computer Engineering

Carlos F. Borges  
Chairman, Department of Applied Mathematics

THIS PAGE INTENTIONALLY LEFT BLANK

## **ABSTRACT**

In Software Defined Radios a good portion (or even the entirety) of the modulation and demodulation processes is performed in the digital domain. The data rate of the transmitted information is very important, since efficiency is a key requirement in real time implementations and cost increases considerably with the number of samples per second to be processed. In this thesis, we address the problem of efficient design of the resampling operations, so that they can be implemented on Field Programmable Gate Arrays (FPGAs).

A set of filtering and resampling operations is developed in the Simulink environment through Xilinx/Simulink blocksets, where all the included subsystems of the design are fully accessible by the designer in any stage of operation. The key ingredient is the use of a Multiplier and Accumulator (MAC) architecture, which can be either time multiplexed for maximum hardware efficiency, or run on a parallel structure for maximum time efficiency.

THIS PAGE INTENTIONALLY LEFT BLANK

# TABLE OF CONTENTS

<b>I.</b>	<b>INTRODUCTION.....</b>	<b>1</b>
<b>A.</b>	<b>BACKGROUND .....</b>	<b>1</b>
	<b>1. FPGA for Digital Signal Processing .....</b>	<b>1</b>
	<b>2. Design Enviroment.....</b>	<b>3</b>
<b>B.</b>	<b>OBJECTIVE .....</b>	<b>4</b>
	<b>1. Efficient Use of the Dual Port Ram and DSP48 Xilinx Blocks .....</b>	<b>5</b>
<b>C.</b>	<b>RELATED WORK .....</b>	<b>8</b>
<b>II.</b>	<b>FINITE IMPULSE RESPONSE FILTER WITH ONE MAC (MULTIPLIER ACCUMULATOR).....</b>	<b>9</b>
<b>A.</b>	<b>THEORETICAL PERSPECTIVE.....</b>	<b>9</b>
<b>B.</b>	<b>SOFTWARE IMPLEMENTATION .....</b>	<b>10</b>
	<b>1. Control Logic for Data and Filter Coefficients .....</b>	<b>11</b>
	<b>2. Alignment of Data and Filter Coefficients in the Dual Port Ram .....</b>	<b>14</b>
	<b>3. Sequential Multiplication and Accumulation (MAC) of Data and Filter Coefficients using DSP48.....</b>	<b>15</b>
<b>C.</b>	<b>RESULTS .....</b>	<b>17</b>
<b>III.</b>	<b>DECIMATION BY AN INTEGER FACTOR.....</b>	<b>21</b>
<b>A.</b>	<b>THEORETICAL PERSPECTIVE.....</b>	<b>21</b>
	<b>1 Sampling Continuous Time Signals .....</b>	<b>21</b>
	<b>2. Analysis of Downsampling (Decimation) .....</b>	<b>22</b>
	<b>3. Efficient Implementation of Decimation Operation using Noble Identities and Filter’s Polyphase Decomposition .....</b>	<b>25</b>
<b>B.</b>	<b>DECIMATION BY TWO WITH FIR MAC AND POLYPHASE DECOMPOSITION.....</b>	<b>26</b>
	<b>1. Software Implementation.....</b>	<b>27</b>
	<i>a. Control Logic for Data and Filter Coefficients .....</i>	<i>29</i>
	<i>b. Sequential Multiplication and Accumulation (MAC) of Data and Filter Coefficients using the DSP48 .....</i>	<i>32</i>
	<i>c. Results.....</i>	<i>34</i>
<b>C.</b>	<b>DECIMATION BY AN INTEGER FACTOR ‘D’ WITH FIR MAC AND POLYPHASE DECOMPOSITION.....</b>	<b>36</b>
<b>IV.</b>	<b>INTERPOLATION BY AN INTEGER FACTOR.....</b>	<b>41</b>
<b>A.</b>	<b>THEORETICAL PERSPECTIVE.....</b>	<b>41</b>
	<b>1. Analysis of Upsampling (Interpolation).....</b>	<b>41</b>
	<b>2. Efficient Implementation of Interpolation Operation using Noble Identities and Filter’s Polyphase Decomposition .....</b>	<b>43</b>
<b>B.</b>	<b>INTERPOLATION BY TWO WITH FIR MAC AND POLYPHASE DECOMPOSITION.....</b>	<b>45</b>
	<b>1. Software Implementation.....</b>	<b>45</b>
	<i>a. Control Logic for Data and Filter Coefficients .....</i>	<i>47</i>

<i>b.</i>	<i>Sequential Multiplication and Accumulation (MAC) of Data and Filter Coefficients using DSP48.....</i>	<i>50</i>
<i>c.</i>	<i>Results.....</i>	<i>52</i>
<b>C.</b>	<b>INTERPOLATION BY AN INTEGER FACTOR ‘D’ WITH FIR MAC AND POLYPHASE DECOMPOSITION .....</b>	<b>53</b>
<b>V.</b>	<b>CONCLUSIONS .....</b>	<b>59</b>
<b>A.</b>	<b>SUMMARY OF THE WORK .....</b>	<b>59</b>
<b>B.</b>	<b>SUGGESTION FOR FUTURE WORK .....</b>	<b>60</b>
	<b>LIST OF REFERENCES .....</b>	<b>61</b>
	<b>INITIAL DISTRIBUTION LIST .....</b>	<b>63</b>

## LIST OF FIGURES

Figure 1.	Actual view of FPGA VIRTEX 4 (From: [4]).	2
Figure 2.	Physical view of FPGA VIRTEX-4 (From: [2]).	2
Figure 3.	Simulink environment using Xilinx.	3
Figure 4.	Basic Structure of Simulation.	4
Figure 5.	Dual Port Ram Xilinx Block.	5
Figure 6.	DSP48 Slice (From: [6]).	7
Figure 7.	DSP48 Xilinx Block.	7
Figure 8.	Discrete Convolution.	9
Figure 9.	Finite Impulse Response Filter with One MAC.	11
Figure 10.	Controller.	13
Figure 11.	Time Representation of Simulation.	14
Figure 12.	Outcome of Dual Port Ram.	15
Figure 13.	Outcome of DSP48.	16
Figure 14.	Frequency Spectrum of the Original and Filtered Signal (Sinusoidal Case).	18
Figure 15.	Frequency Spectrum of the Original and Filtered Signal (Gaussian White Noise Case).	19
Figure 16.	Sampling Continuous Time Signals.	22
Figure 17.	Downsampling Operation.	23
Figure 18.	Aliasing Effect in Frequency Spectrum.	24
Figure 19.	Filtering and Downsampling a Discrete Signal.	25
Figure 20.	Efficient Implementation of Decimation.	26
Figure 21.	Downsampling by Two.	28
Figure 22.	Controller.	30
Figure 23.	Time Representation of Simulation.	31
Figure 24.	Outcome of Dual Port Ram.	32
Figure 25.	Outcome of DSP48.	33
Figure 26.	Frequency Spectrum of the Original and Downsampled by Two Signal without Aliasing Effect.	35
Figure 27.	Frequency Spectrum of the Original and Downsampled by Two Signal with Aliasing Effect.	36
Figure 28.	Timing Diagram for Decimation by D.	38
Figure 29.	Frequency Spectrum of the Original and Downsampled by $D = 4$ Signal.	40
Figure 30.	Upsampling Operation.	41
Figure 31.	'Ghost' Frequencies in Interpolation Operation.	42
Figure 32.	Upsampling and Filtering with LPF.	43
Figure 33.	Efficient Implementation of Interpolation.	44
Figure 34.	Upsampling by Two.	46
Figure 35.	Controller.	48
Figure 36.	Time Representation of Simulation.	49
Figure 37.	Outcome of Dual Port Ram.	50
Figure 38.	Outcome of DSP48.	51
Figure 39.	Frequency Spectrum of the Original and Upsampled by Two Signal.	53

Figure 40.	Timing Diagram for Interpolation by $D$ .....	55
Figure 41.	Frequency Spectrum of the Original and Upsampled by $D = 4$ Signal. ....	57

## EXECUTIVE SUMMARY

In Software Defined Radios (SDR) a good portion (or even the entirety) of the modulation and demodulation process is performed in the digital domain. The reconfigurability and the versatility of the SDR can be efficiently supported by the Field Programmable Gate Arrays (FPGAs) for hardware implementations.

FPGAs are high performance integrated circuits suitable for many Digital Signal Processing (DSP) applications with the feature of being reprogrammable by the designer. In this way, the system can be easily reconfigured to a number of different applications.

The proper software needed to program an FPGA is provided by System Generator (Sysgen), which is an FPGA design program responsible for driving the FPGA through the high-level design environment of Simulink. A combination of common and synthesized Simulink/Xilinx blocks from the Simulink library along with MATLAB codes have been used in order to construct a configurable scheme capable of implementing the following three operations:

- a) Finite Impulse Response (FIR) filter
- b) Decimation by an integer factor
- c) Interpolation by an integer factor

The key ingredient is the use of the Multiplier and Accumulator (MAC) architecture, which can be either time multiplexed for maximum hardware efficiency, or embedded on a parallel structure for maximum time efficiency.

The main components of the implementation are the Dual Port Ram Xilinx block, which is a random access memory containing both data and the FIR filter coefficients, together with the DSP48 Xilinx block, which performs the multiplication and addition on a sequential basis. The DSP48 block is specifically designed for high-speed arithmetic operations and it is part of the standard Xilinx Virtex family architecture. The objective is to perform the proper arrangement of the input data and FIR filter coefficients so that the resulting multiplication and accumulation will perform the three examined operations

according to the theoretical formulations. Since the operations are performed serially, the data need to be upsampled in order to handle the increased clock rate provided by System Generator (Sysgen) and then properly downsampled.

In this research we have shown that for all three cases (FIR filter, Decimation, Interpolation) the overall structure is the same. What defines each operation is the control logic (Controller) and the storing of the filter parameters.

The controller consists of logic blocks from the Xilinx blockset and it is responsible for updating the Dual Port Ram's memory vectors (according to Sysgen clock rate) in order to provide the proper dual sequential output. The dual output of the memory block is multiplied and accumulated by DSP48 math slice. The outcome of the DSP48 is a bitstream in which the desired coefficient of the three examined operations are embedded accordingly in multiple of the Sysgen rate. Therefore, the final output can be obtained by downsampling the output of DSP48 with the proper factor.

MATLAB was used to verify the consistency of the simulation with the theory.

## ACKNOWLEDGMENTS

Θα ήθελα να ευχαριστήσω και να εκφράσω την βαθιά μου αγάπη στη σύζυγο μου Ελένη και στα παιδιά μου Δημοσθένη και Παναγιώτη που μου συμπαραστάθηκαν καθ'όλη τη διάρκεια των σπουδών και της προσπάθειας ολοκλήρωσης της πτυχιακής, επιδεικνύοντας ιδιαίτερη υπομονή και κατανόηση.

Ringrazio il mio insegnante e relatore Professor Roberto Cristi per la sua disponibilità e il suo sostegno nella preparazione della mia tesi.

I would also like to thank my co-advisor Professor Craig W. Rasmussen for his contribution to accomplish my thesis and the mathematical knowledge he imparted to me.

THIS PAGE INTENTIONALLY LEFT BLANK

# I. INTRODUCTION

## A. BACKGROUND

In Software Defined Radio, the modulation and demodulation processes are performed in the digital domain. The data rate of the transmitted signal is usually several orders of magnitudes smaller than the data rate necessary to drive the Digital to Analog Converters (DACs) at the radio frequency (RF). In real time implementations, since the cost increases according to the number of samples per second, we need to adapt the sampling rate to the frequency content of the transmitted signal. Therefore, signals at radio frequency (RF) are sampled at a rate comparable to the RF frequency, while the signals at baseband are sampled at the information rate [1]. The reconfigurability and the versatility of the SDR can be efficiently supported by the Field Programmable Gate Arrays (FPGAs) for hardware implementations.

### 1. FPGA for Digital Signal Processing

The Field Programmable Gate Array (FPGA) is a high performance integrated circuit suitable for Digital Signal Processing (DSP) applications. An FPGA has the feature of being programmable by the designer and it can be easily reprogrammed. Physically, an FPGA is a two-dimensional array of gates consisting of various logic DSP blocks and interconnections between them in order to perform DSP operations [2].

Figure 1 shows a Virtex-4 FPGA embedded in a processing board. Figure 2 shows a number of important features such as the array of ‘slices’ disposed in columns of macroblocks. The latter are blocks, constituted of memory and arithmetic units that are programmed to perform suitable operations. The entire interconnected mesh can be programmed into highly parallel algorithms [2].

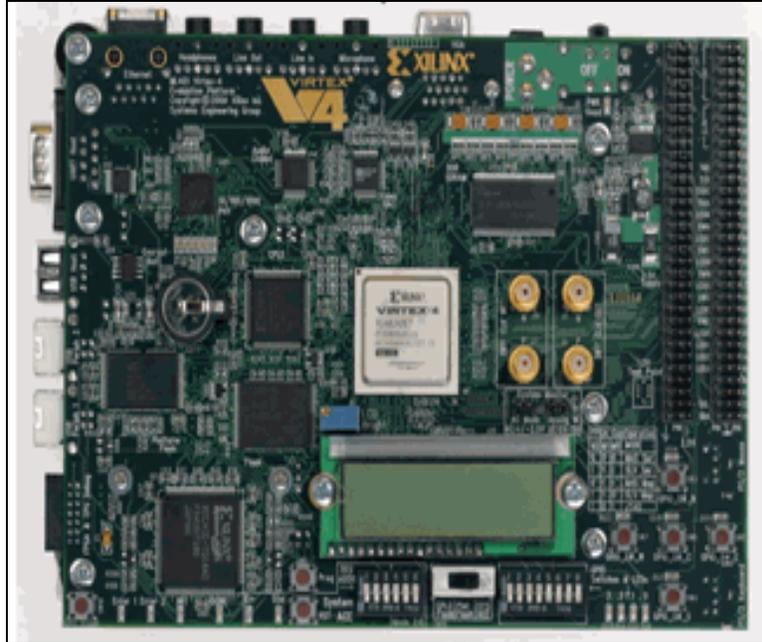


Figure 1. Actual view of FPGA VIRTEX 4 (From: [4]).

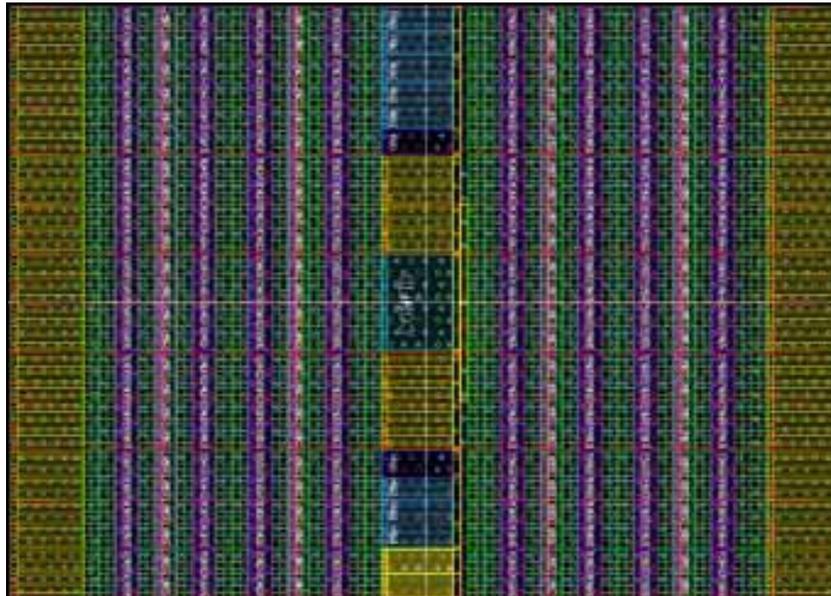


Figure 2. Physical view of FPGA VIRTEX-4 (From: [2]).

## 2. Design Environment

The Xilinx DSP blockset is a suitable tool for designing FPGA algorithms in the Mathworks Simulink design environment. This is supported by the System Generator (Sysgen), which is a FPGA design program responsible for driving the FPGA through the high-level design environment of Simulink. A sufficient number of common and complex blocks, which are provided from several blocksets (including the Xilinx blockset) of the Simulink Library, are properly synthesized in order to design various DSP applications [5]. Figure 3 shows on the left the Simulink Library Browser with various basic elements of the Xilinx blockset, and, on the right of the same figure, a simple application in Simulink using Sysgen. Specifically, an input data sequence is loaded from MATLAB's workspace and upsampled by a factor of two. The output is shown on the 'Scope' by double clicking the corresponding icon. Both 'in' and 'out' blocks are the interfaces of common Simulink blocks with the Xilinx blockset. The entire system is controlled by the Sysgen block. The specified parameters of all blocks can be modified by the user when the respective icon is selected.

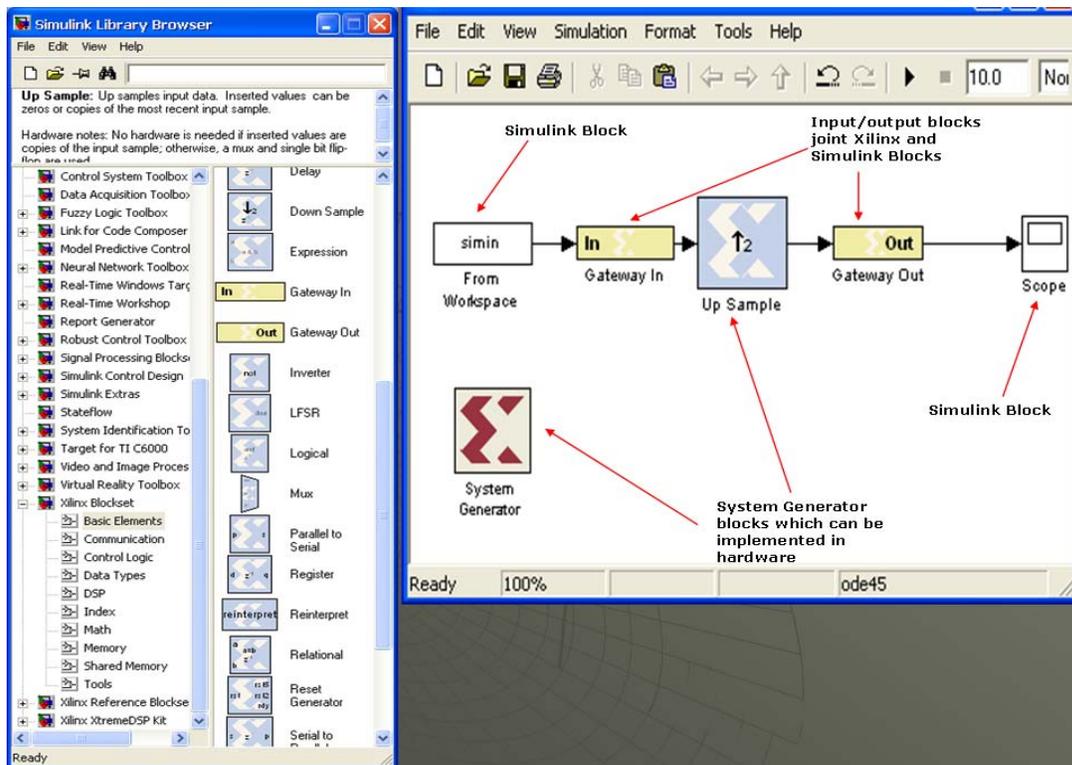


Figure 3. Simulink environment using Xilinx.

## B. OBJECTIVE

In this thesis, we address the problem of efficient design of resampling operations so they can be implemented on Field Programmable Gate Arrays (FPGAs). The key ingredient is the use of a Multiplier and Accumulator (MAC) architecture, which will allow us to perform the following operations:

- 1) Finite Impulse Response (FIR) filters
- 2) Decimation by an integer factor
- 3) Interpolation by an integer factor

The outcome of these three schemes is the development of a set of filtering and resampling operations performed in Xilinx/Simulink. All the subsystems in the designs are fully accessible by the designer.

In order to perform the three operations (FIR filtering, Decimation and Interpolation by an integer factor), a basic design scheme in the Simulink environment is used and is modified accordingly to fit the three cases. Since the objective is to develop software suitable to programming FPGAs, a combination of Xilinx and Simulink blocks as well as MATLAB codes is used. Figure 4 illustrates the basic structure of the Simulation.

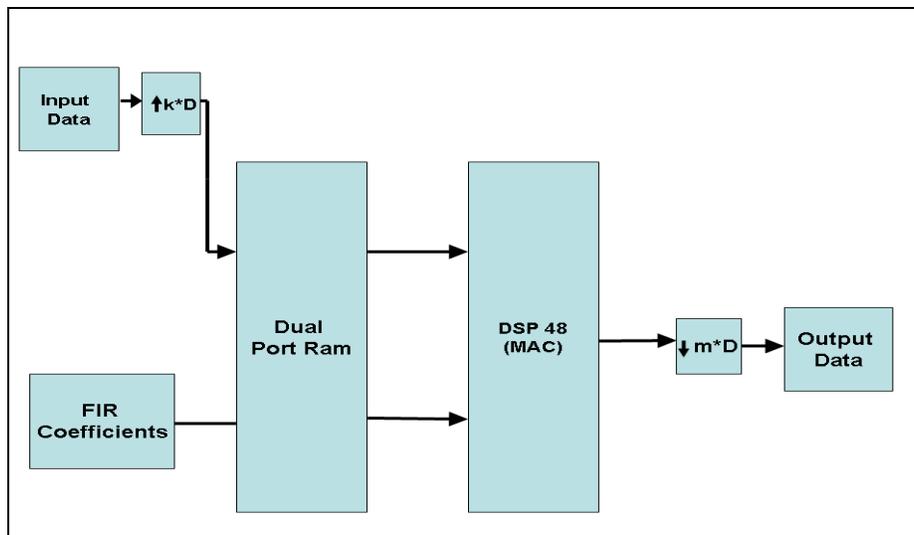


Figure 4. Basic Structure of Simulation.

In each of the three designs, the proper arrangement of the input data points and Finite Impulse Response (FIR) filter coefficients is achieved in the Dual Port Ram Xilinx block, which is a random access memory. The dual output of the memory block is multiplied and accumulated by the DSP48 Xilinx block, which is an efficient block for DSP operations implementing a Multiplier and Accumulator (MAC) operation. From the resulting output, we selectively extract the data points of interest according to the theoretical formulas of the three desired operations. Although several Xilinx/Simulink blocks are used and are explained in the next chapters, the principal blocks are the Dual Port Ram and the DSP48.

### 1. Efficient Use of the Dual Port Ram and DSP48 Xilinx Blocks

The Dual Port Ram Xilinx block is a dual memory device that allows the user to specify the width and the values of each memory part. This specific block uses two sets of ports dedicated to reading and writing of data. Each port has three inputs: (a) the address line 'addr', (b) the input data 'din' and (c) the write enable 'we'. In addition, each port has one output. There is also an option of additional enable and synchronous reset inputs for both ports that were not necessary for the purpose of this design. The Dual Port Ram Xilinx block, along with its specified parameter window, is shown in Figure 5.

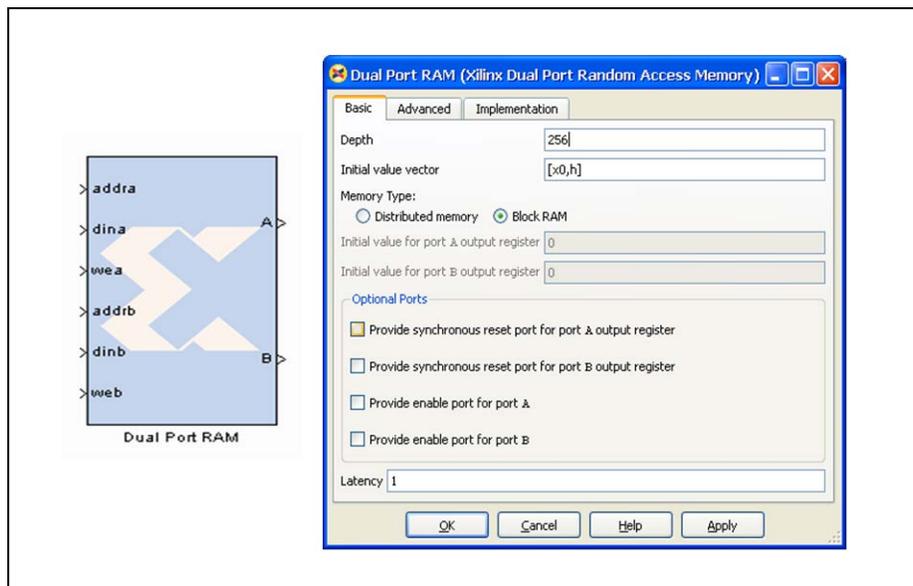


Figure 5. Dual Port Ram Xilinx Block.

Both memories are accessible for reading and writing by providing the right address from the address ports 'addra' and 'addrb'. The initial value vector, as it is indicated in the parameter window of Figure 5, is the concatenation of the two initial vectors (initial data vector  $x_0$  and initial FIR filter coefficients  $h$ ). The 'wea' and 'web' are the write enable ports for each memory feeding the Dual Port Ram with a Boolean signal '0' or '1'. When the 'we' port is set to 1 then the memory writes the value of the 'din' port to the location specified by the corresponding address line. Each of the two outputs depend on the write mode, which in our case is 'read after write', and it takes exactly the same value indicted by the address line when the write cycle is completed [5].

For the purposes of this thesis, the second part of the memory remains unchanged (no input data) and keeps the initial value. Specifically, input  $b$  takes the values of properly ordered (according to the case of interest) finite impulse response (FIR) filter coefficients, which are generated in the initialization of the simulation through any MATLAB function such as 'firpm'. Therefore ports 'dinb' and 'web' are fed with a signed and a boolean zero respectively. On the other hand, the first part of the memory changes according to 'address' and 'write enable' ports.

The outputs of ports  $A$  and  $B$  are two signed bit streams: one for the input data points and one for coefficients of the FIR filter, aligned in such a way so that their multiplication and accumulation will provide us the desired result for the three examined cases.

The DSP48 Xilinx block (also referred as an extreme DSP slice or DSP48 math slice) is an efficient tool for many DSP applications, which can handle dynamically many operations as well as be cascaded with other DSP48 blocks. It consists of an 18-bit-by-18-bit signed multiplier with a 48-bit adder and a programmable multiplexer that can be driven as required to perform specific operations [3]. The logic circuit of the slice is depicted in Figure 6, while the corresponding Xilinx block along with some capable operations is shown in Figure 7.

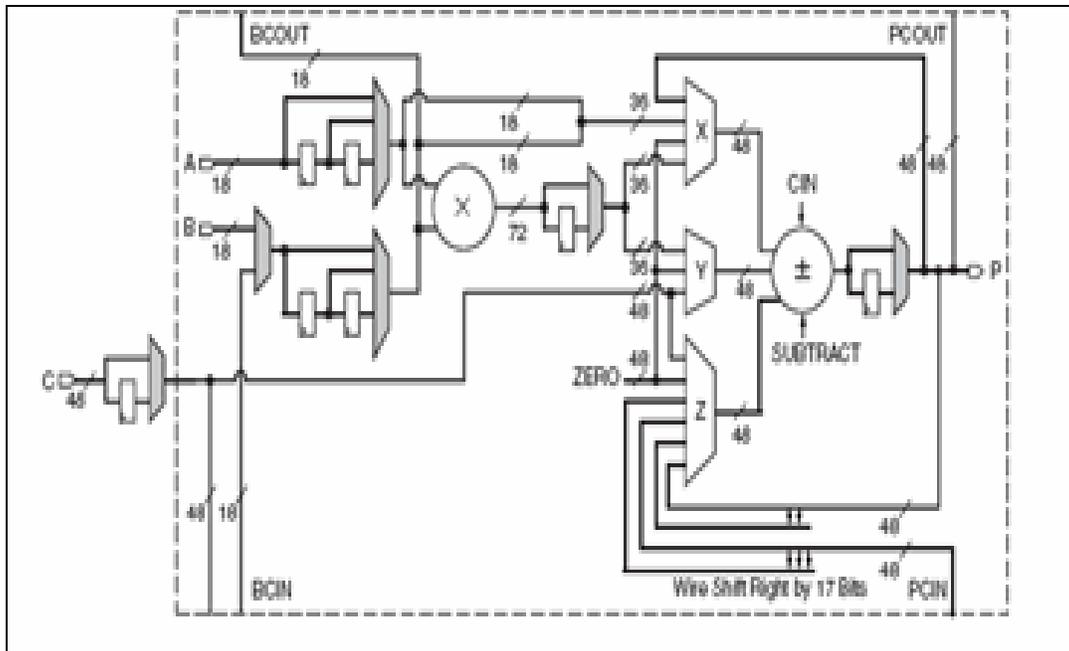


Figure 6. DSP48 Slice (From: [6]).

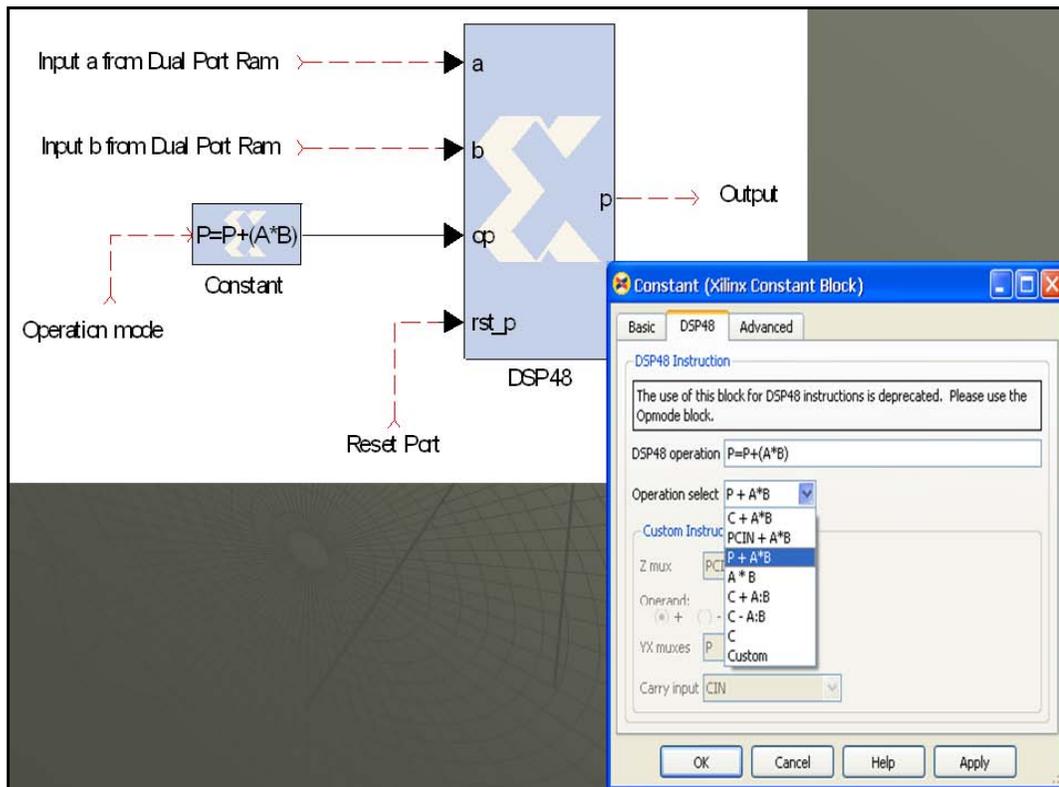


Figure 7. DSP48 Xilinx Block.

In this thesis the DSP48 is used as a multiplier and accumulator (MAC) block and its operation is defined as  $P = P + A \cdot B$ . With this block the product of two inputs  $A$  and  $B$  (derived from the Dual Port Ram) is accumulated each time with the previous product  $P$ . A reset port is available to the slice in order to reset the output every clock cycle to produce the desired for each examined case operation.

### **C. RELATED WORK**

Although a number of approaches to FIR filtering and resampling operations design exist in literature ([10], [11]), to the best knowledge of the author there has been no systematic way of designing these filters in a general fashion.

The main contribution of this resurch is an architecture, which is fully scalable to any implementation in terms of filter coefficients and resampling factor.

## II. FINITE IMPULSE RESPONSE FILTER WITH ONE MAC (MULTIPLIER ACCUMULATOR)

### A. THEORETICAL PERSPECTIVE

In the digital domain, the output sequence  $y[n]$  of a Finite Impulse Response (FIR) filter is given by the following expression:

$$y[n] = \sum_{k=0}^N h[k] \cdot x[n-k], \quad (2.1)$$

where  $h[n]$  is the impulse response of the filter,  $x[n]$  is the input sequence and  $N$  being the degree of the transfer function of the FIR filter.

Both  $x[n]$  and  $y[n]$  are at the same clock rate  $F_x = F_y = F_s$  as  $x[n] = x(nT_s)$  and  $y[n] = y(nT_s)$ , where  $T_s = \frac{1}{F_s}$  is the sampling interval [7]. The discrete convolution, along with its graphical representation, is depicted in Figure 8.

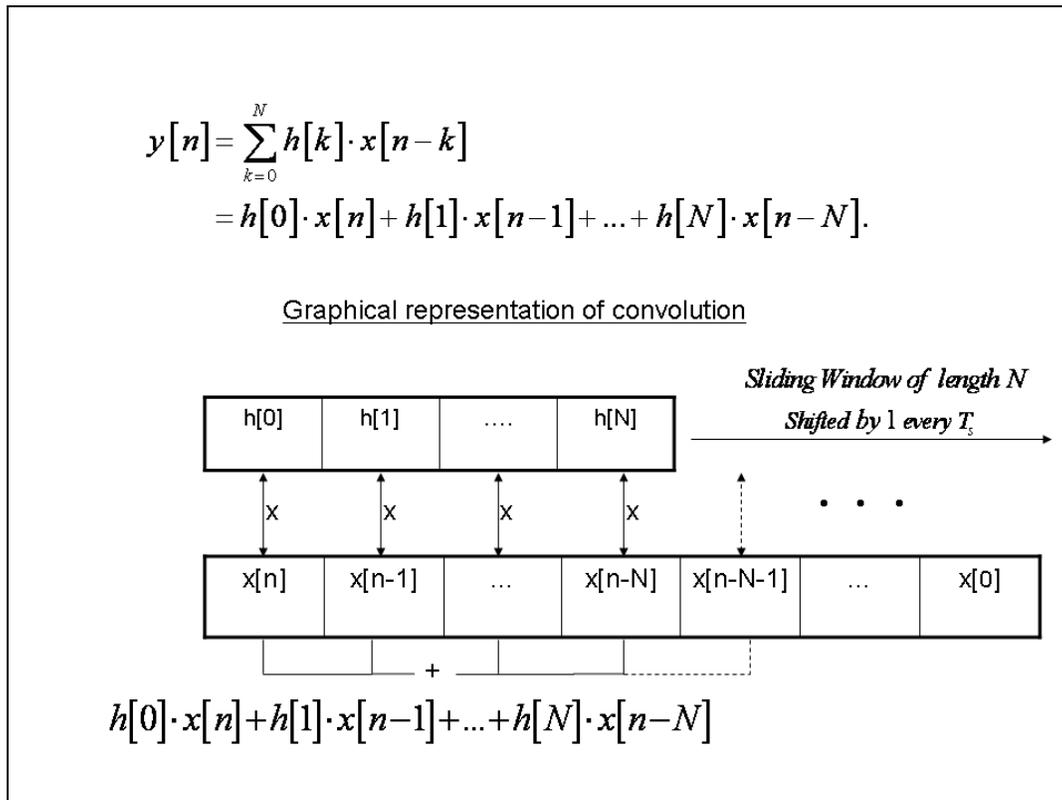


Figure 8. Discrete Convolution.

We can verify from Figure 8 that the convolution operation can be graphically implemented as a sliding window over a data sequence. In particular, at any time  $n$  we need to save  $N+1$  data points  $x[n], x[n-1], \dots, x[n-N]$  together with the coefficients  $h[0], h[1], \dots, h[N]$ .

In this chapter, we address the problem of implementing the filtering operation using one Multiplier and Accumulator (MAC). In this way, the convolution sum is computed in about  $N$  clock pulses (where  $N$  denotes the degree of the transfer function of the FIR filter), thus requiring a higher clock rate to be provided by the System Generator, which controls the operation and its parameters. The objective is to perform the proper arrangement of the input data points and the filter's coefficients so that the multiplication and accumulation procedure as well as the selective extraction of outcomes will give us the desired convolution result in the most efficient way.

## **B. SOFTWARE IMPLEMENTATION**

The Simulink/Xilinx implementation needed to perform the FIR filtering is shown in Figure 9. The main components of the implementation are the Dual Port Ram, which contains both data and the FIR filter coefficients and the DSP48, which performs the multiplication and addition on a sequential basis. Since the operations are performed serially, the data need to be upsampled in order to handle the increase of the clock rate provided by System Generator. The controller consists of a set of counters (one for the coefficients and one for the data points) along with logic blocks (implemented in Xilinx blockset), and controls the flow of the data at the output of the dual Port Ram as well as the timing of the operations.

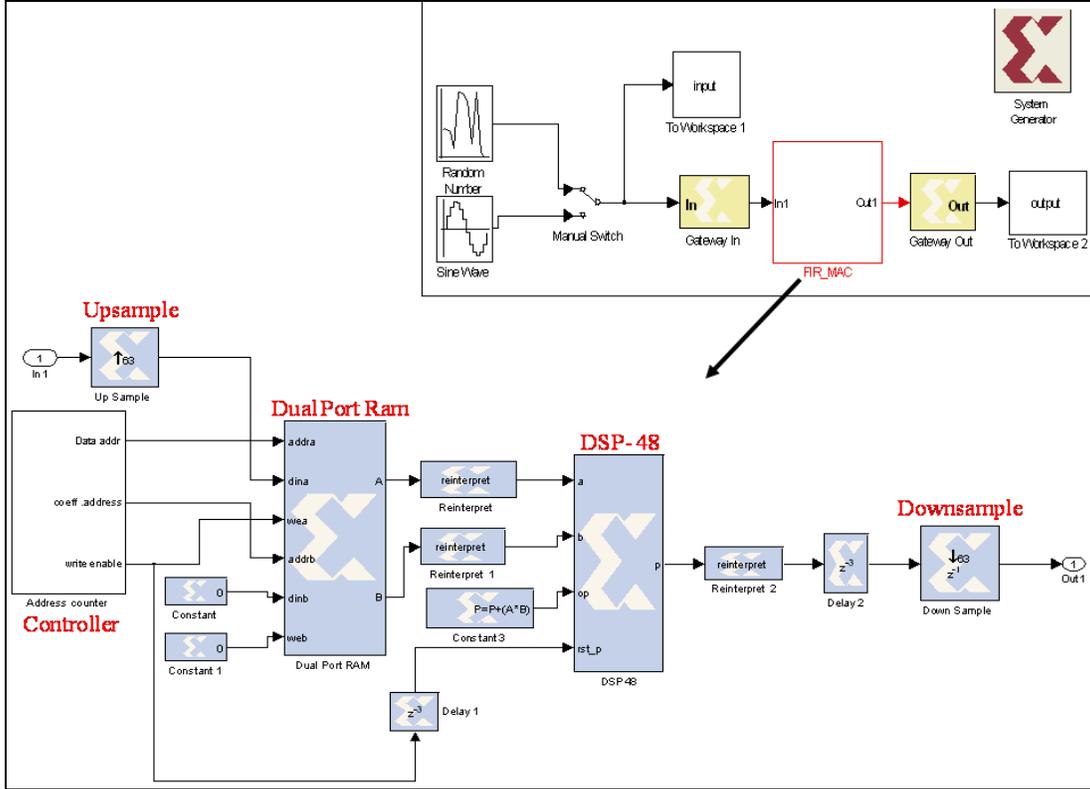


Figure 9. Finite Impulse Response Filter with One MAC.

In order to test the performance of the filter, a Gaussian white noise and a sinusoidal signal are selectively available (by a manual switch) as inputs. The input signal is sampled at rate  $F_s$ , while System Generator (Sysgen) works at a higher sampling rate equal to  $(N + 1)F_s$ . Since the new system rate provided by Sysgen is higher, the input data is upsampled by the integer factor of  $N + 1$  with the corresponding Xilinx block.

The objective is to achieve a proper alignment of the data and filter's coefficients, so that they can be applied to a MAC resulting in the convolution operation. Towards this goal, we need two memory vectors  $x$  and  $h$  containing the data and the filter coefficients respectively provided by the Dual Port Ram and a MAC provided by the DSP48.

### 1. Control Logic for Data and Filter Coefficients

The vector  $h$  of the filter coefficients is defined as  $h = [h[0], h[1], h[2], \dots, h[N-1], 0]$ . It has length  $N + 1$  and it remains unchanged during

the operation of the filter. Therefore, the ports ‘dinb’ (data input b) and ‘web’ (write enable b) are set to false. The first  $N$  coefficients of the vector  $h$  are generated in MATLAB as an FIR filter using function ‘firpm’, while the additional  $(N+1)^{th}$  coefficient is intentionally set to zero in order to serve computational issues derived from the use of the DSP48, which works as a MAC and will be explained in the MAC procedure.

The input data vector stored in the first part of the memory of the Dual Port Ram is a circular shift register of length  $N$ , updated at times  $t = nT_s$  by  $X[(n)_N] \leftarrow x[n]$ ,  $t = nT_s$ , with  $(n)_N = 0, 1, \dots, N-1$  denoting modulo operation. In the implementation,  $(n)_N$  is a periodic counter with update rate  $F_{ac} = (N+1)F_s$ . The initial value of the memory vector  $x$  is set to the initial conditions (say zero for example) and updates its value according to the corresponding ‘address’ and ‘write enable’ ports provided from the controller. Figure 10 illustrates the controller of this design.

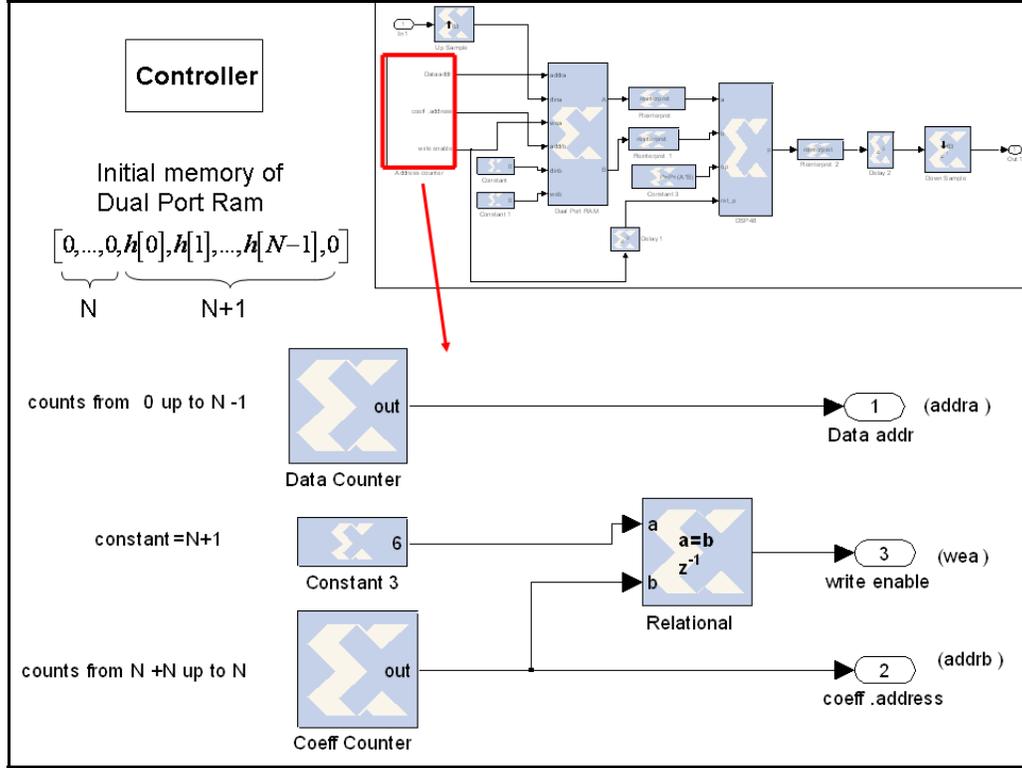


Figure 10. Controller.

The time representation of ‘data address’ and ‘coefficients address’ sequences of Figure 10 is shown in Figure 11. In particular, at time  $(n-1)T_s$  the accumulator is initialized by  $a((n-1)T_s) = 0$  (where ‘a’ denotes the content of the accumulation). At every subsequent clock cycle  $T_{ac} = \frac{1}{F_{ac}}$  the accumulation will be updated as

$$a((n-1)T_s + \lambda T_{ac}) = a((n-1)T_s + (\lambda-1)T_{ac}) + (data\_addr(\lambda) \cdot coeff\_addr(\lambda)),$$

where  $\lambda = 1, \dots, N$ . The output  $y[n]$  at time  $nT_s$  is shown in the timing diagram of Figure 11.

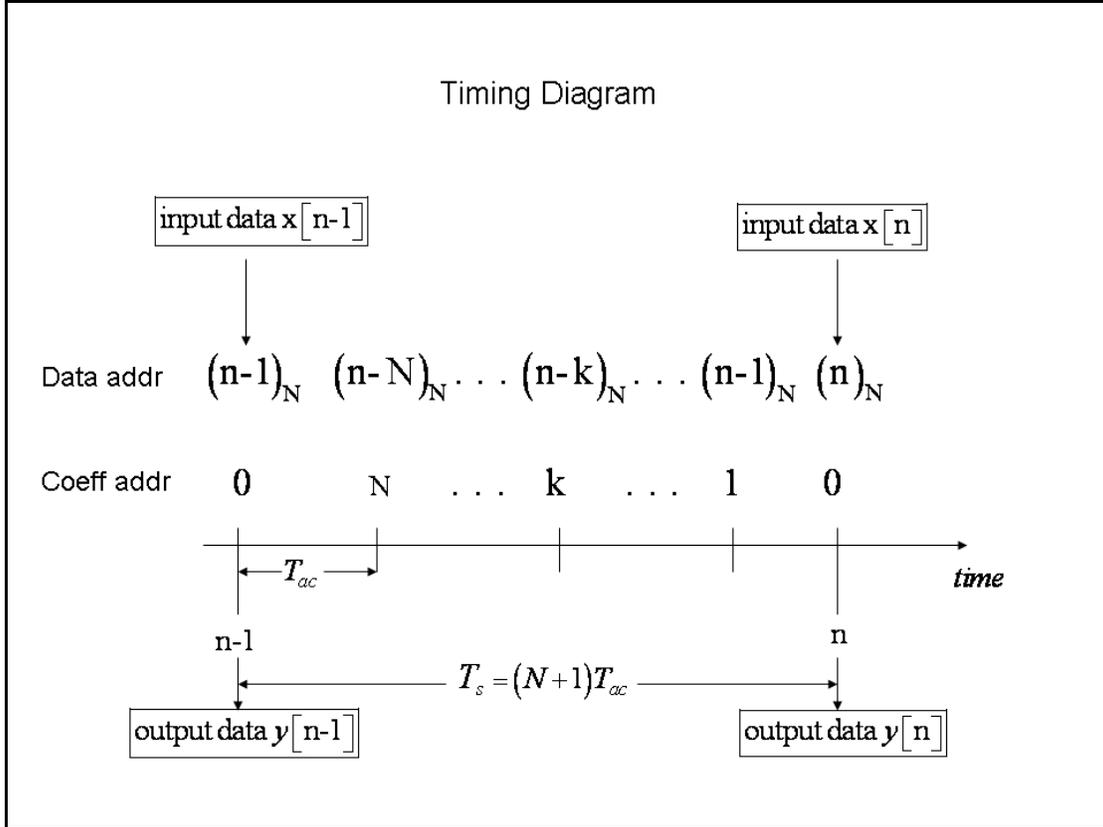


Figure 11. Time Representation of Simulation.

In what follows we demonstrate the functionality of the design according to the timing diagram illustrated in Figure 11.

## 2. Alignment of Data and Filter Coefficients in the Dual Port Ram

The length of the vector  $x$  is chosen to be one less than the length of  $h$  so that the writing procedure will introduce a shift by a factor of one in the content of memory  $x$ . It can be inferred that the outcome of the Dual Port Ram is a set of bitstreams, where the output at port  $A$  is a recurrent window of length  $N+1$  (in every  $T_s$ ) in which the input data is progressively shifted by one position from left to right, while the bitstream of port  $B$  is a repetition of the vector  $h$ . Figure 12 illustrates the outcome of the Dual Port Ram with time running from right to left.

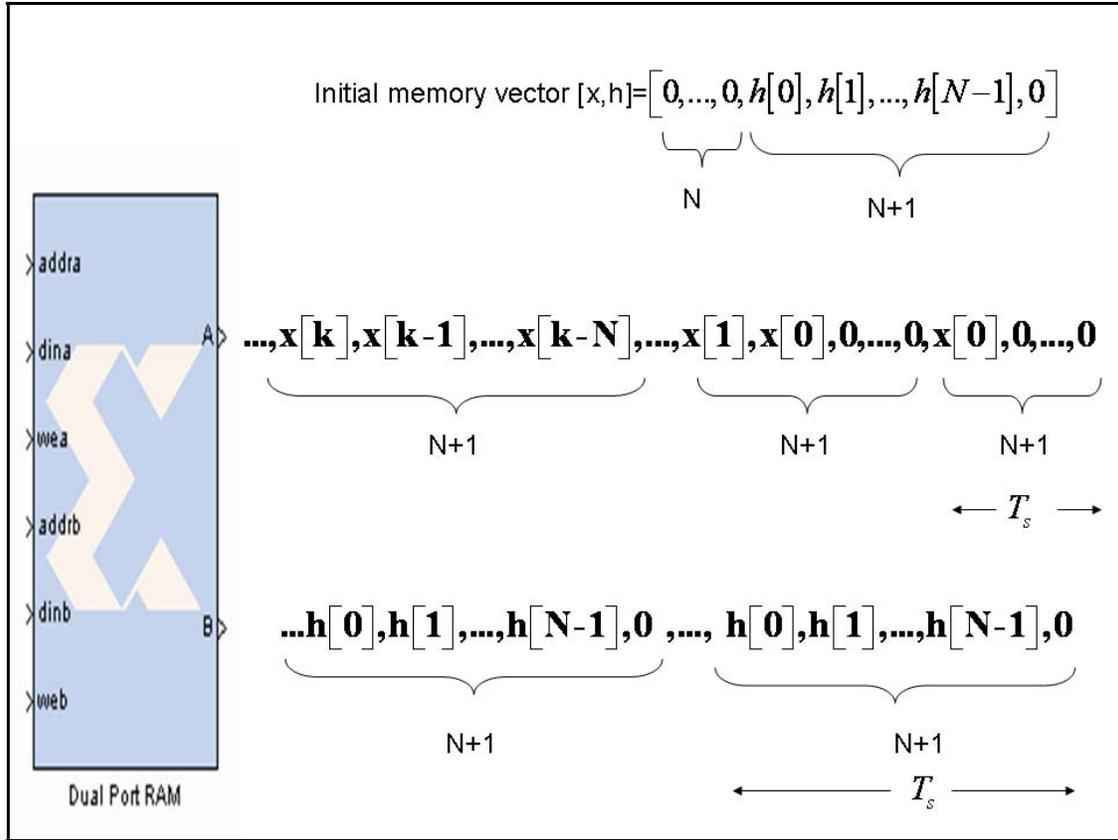


Figure 12. Outcome of Dual Port Ram.

### 3. Sequential Multiplication and Accumulation (MAC) of Data and Filter Coefficients using DSP48

The output bitstream from the Dual Port Ram, as shown in Figure 12, is being processed by the DSP48 Xilinx block, which works as a MAC. Its operation mode is defined as  $P = P + A \cdot B$  (referring to Figure 7) where the product of two pairs of the Dual Port Ram output ports  $A$  and  $B$  is being accumulated each time with the previous product. A reset signal (selected from the DSP48 options) for the outcome  $P$  is introduced at clock rate  $N+1$  provided from the properly delayed 'write enable' signal of the controller of the Dual Port Ram (referring to Figure 9). The adjustment of the delay is set so that the reset of the outcome  $P$  occurs every  $N+1$  times, where a data coefficient is multiplied with the zero coefficient of vector  $h$ .

Consequently, considering the length  $(N+1)$  of block pairs in Figure 12, whenever a product  $x[k] \cdot h[0]$  (with  $k$  arbitrarily chosen) is accumulated to the previous  $N$  sums of products of each block pairs, a data point of the convolution  $y[n]$  is produced as shown in the timing diagram (Figure 11). For illustration, Figure 13 shows the first two points  $y[0]$ ,  $y[1]$  computed at times  $(N+1)T_{ac}$  and  $2(N+1)T_{ac}$ , respectively, by the first two sets of blocks.

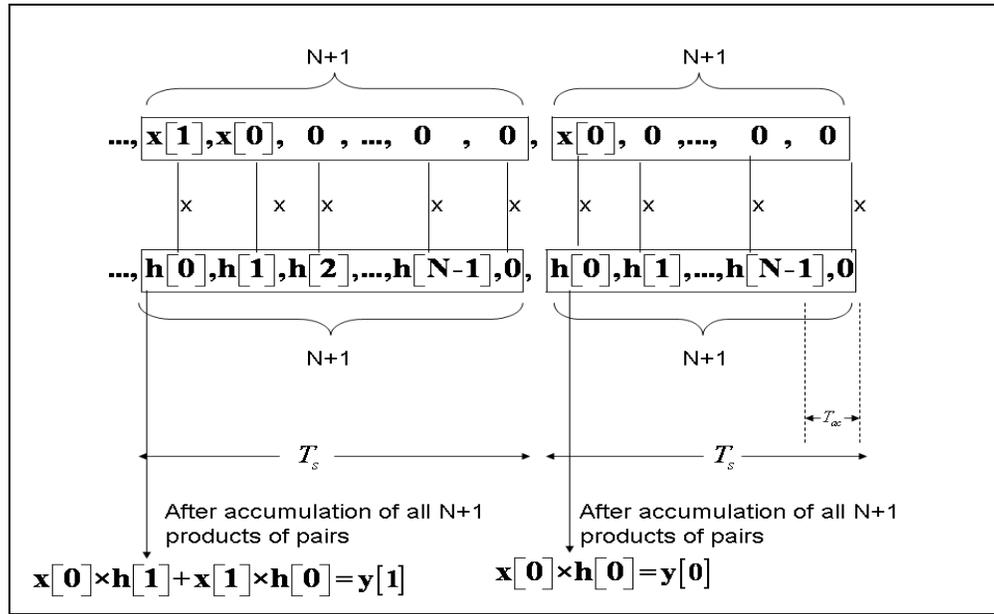


Figure 13. Outcome of DSP48.

Referring to figure, the bitstream outcome  $P$  of the DSP48 can be considered as a set of blocks of length  $N+1$  in which the desired convolution coefficients are embedded in every  $(N+1)^{th}$  element of each block as it shown in the timing diagram in Figure 11. Therefore, by downsampling the data  $P$  by the factor of  $N+1$  (same factor that was used when the input data was upsampled) the desired convolution result is provided.

### C. RESULTS

In order to test the performance, an FIR filter was designed and tested with two classes of input signals. In particular the FIR filter has been designed as an Equiripple Filter with the following characteristics:

Passband: 0-0.2 (in terms of Digital Frequency  $f$  )

Stopband: 0.3-0.5 (in terms of Digital Frequency  $f$  )

Order: 60

The signals tested are a sinusoid and a white noise. The sinusoid has frequency  $F = 0.1 \cdot F_s$  (Hz) with sampling frequency  $F_s = 10000$  (Hz) and  $f = F / F_s$ , while the white noise is sampled at the same rate.

The frequency spectrum of the original signal and the resulting filtered signal for the sinusoidal case is shown in Figure 14. We can verify that the frequency spectrum of the original signal remains the same as long as its frequency is within the passband of the FIR filter.

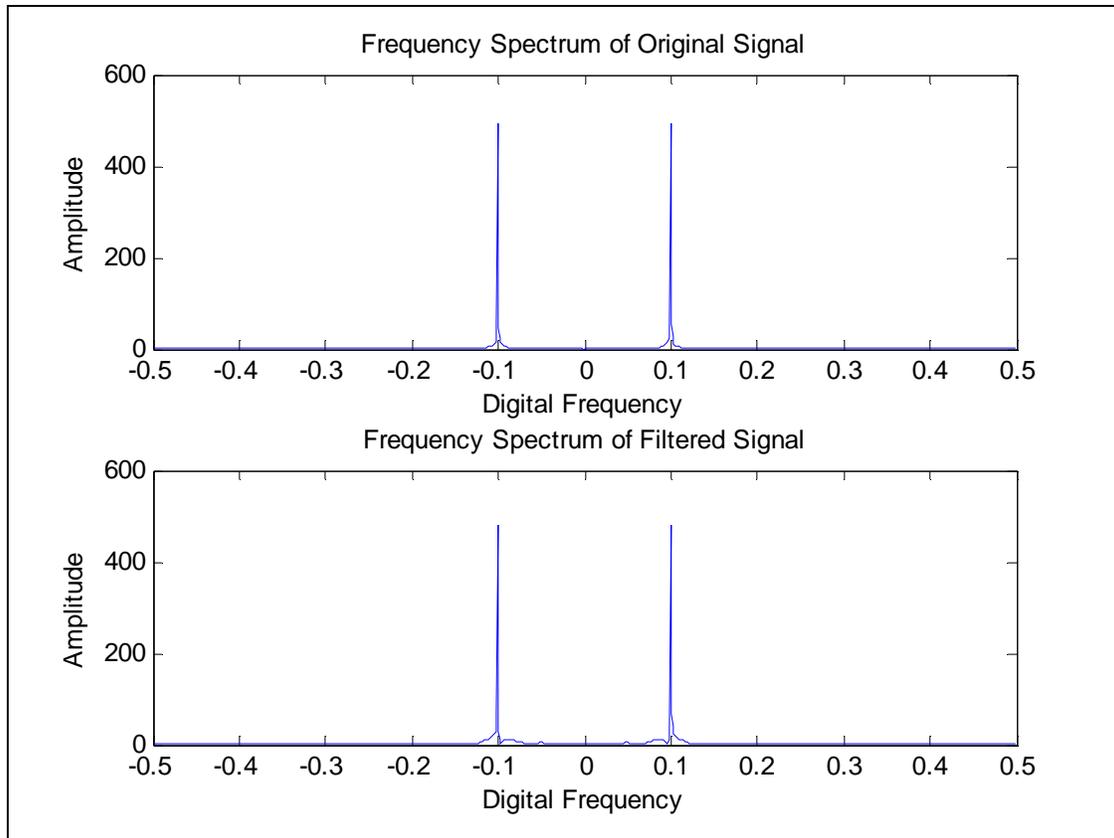


Figure 14. Frequency Spectrum of the Original and Filtered Signal (Sinusoidal Case).

For the Gaussian white noise case the corresponding frequency spectrum, along with the frequency spectrum of the filtered signal, is depicted in Figure 15. We can observe that the frequencies of the Gaussian white noise are spread all over the frequency spectrum while the frequency spectrum of the corresponding filtered signal maintains the frequencies that are within the passband of the FIR filter and eliminates all the others.

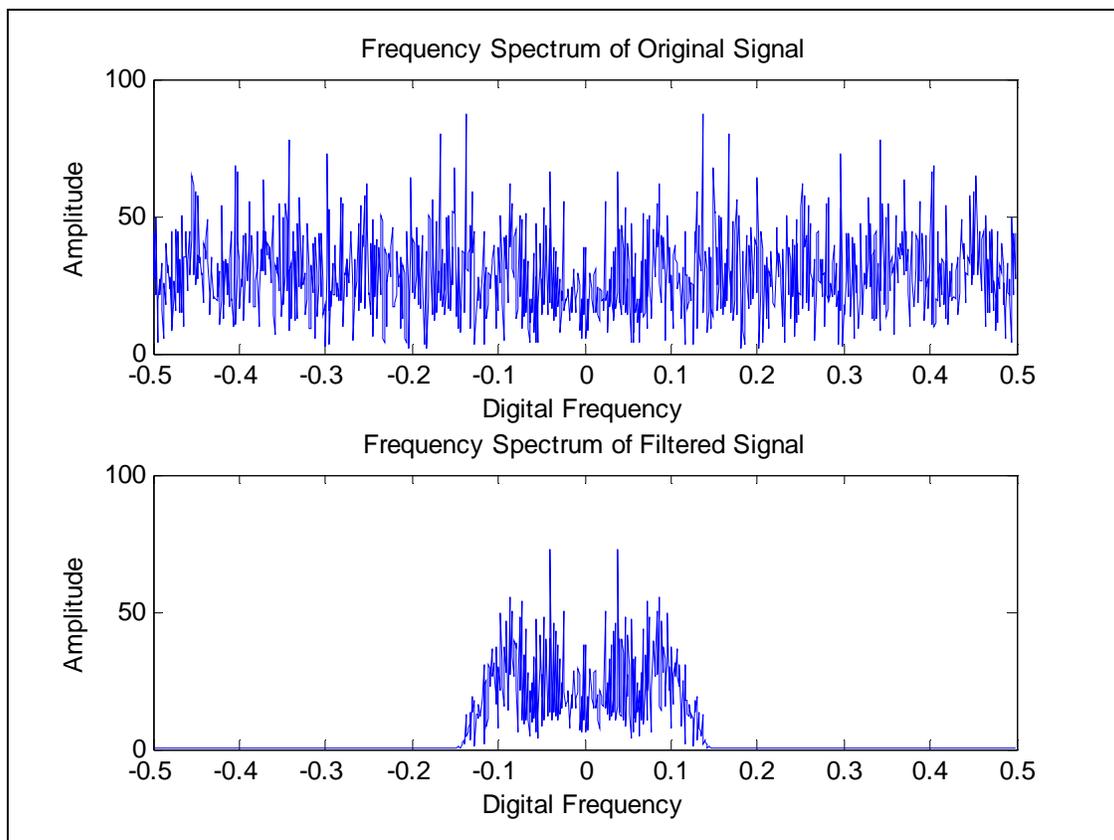


Figure 15. Frequency Spectrum of the Original and Filtered Signal (Gaussian White Noise Case).

THIS PAGE INTENTIONALLY LEFT BLANK

### III. DECIMATION BY AN INTEGER FACTOR

#### A. THEORETICAL PERSPECTIVE

##### 1 Sampling Continuous Time Signals

It is well known that by the sampling theorem, the sampling frequency  $F_s$  has to be at least twice the signal bandwidth  $B$  [7]. The Discrete Time Fourier Transform of a sampled signal  $x[n]$  with actual frequency content  $F$ , which is sampled at rate  $F_s$ , is given by the following expression:

$$X(f) = DTFT \{x[n]\} = \sum_{i=-\infty}^{+\infty} x[n] e^{-j2\pi fn}, \quad (3.1)$$

where  $f$  is a dimensionless quantity denoting the digital frequency  $f = \frac{F}{F_s}$ . From

equation (3.1) we can verify that  $X(f)$  is periodic with period one since

$$X(f+1) = \sum_{n=-\infty}^{+\infty} x[n] e^{-j2\pi(f+1)n} = \sum_{n=-\infty}^{+\infty} x[n] e^{-j2\pi fn} = X(f).$$

Therefore, the information is contained in one period (within the interval  $-1/2 \leq f \leq 1/2$ ) of the periodic repetition of the frequency spectrum. Figure 16 illustrates the frequency spectrum of a continuous time and sampled signal respectively.

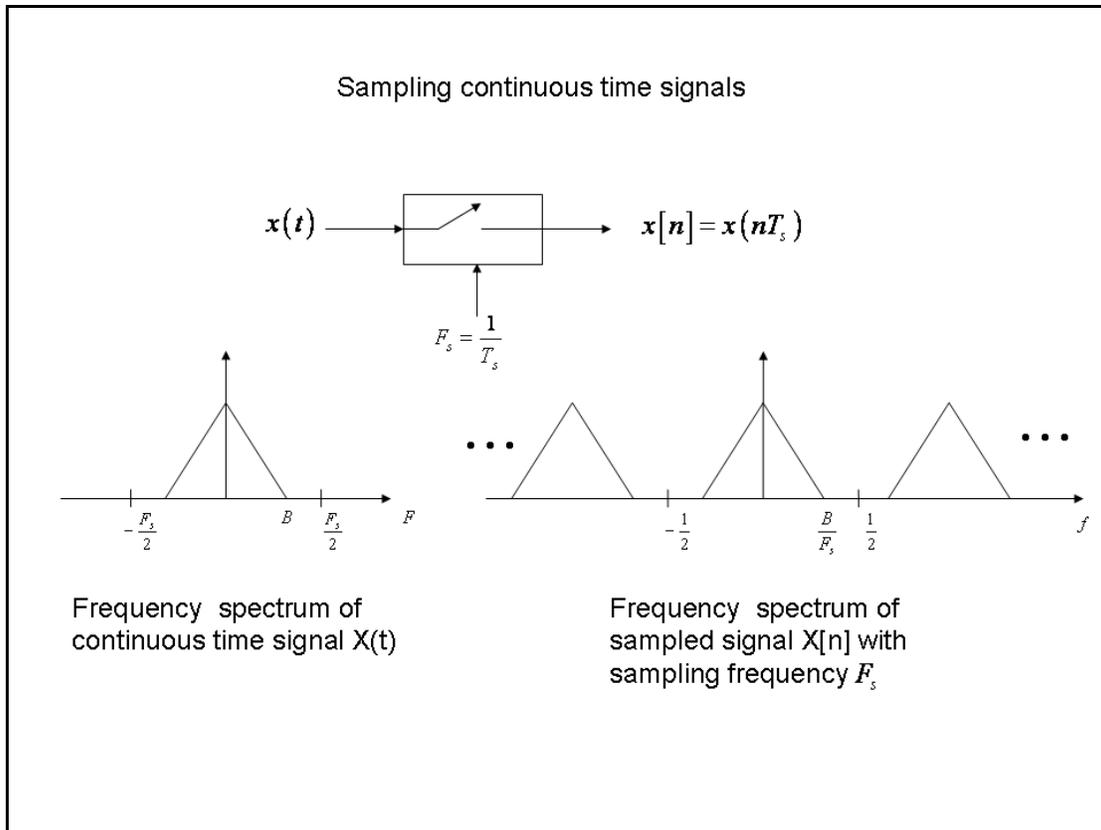


Figure 16. Sampling Continuous Time Signals.

## 2. Analysis of Downsampling (Decimation)

In digital communications such as Software Defined Radio, the exchange of information needs to be done in the most efficient way, in order to reduce complexity and improve efficiency while preserving the content of the information. The Downsampling operation (Decimation) decreases the number of samples per second of a given signal by an integer factor of  $D$ . An example of decimation by integer factor of  $D = 3$  is shown in Figure 17.

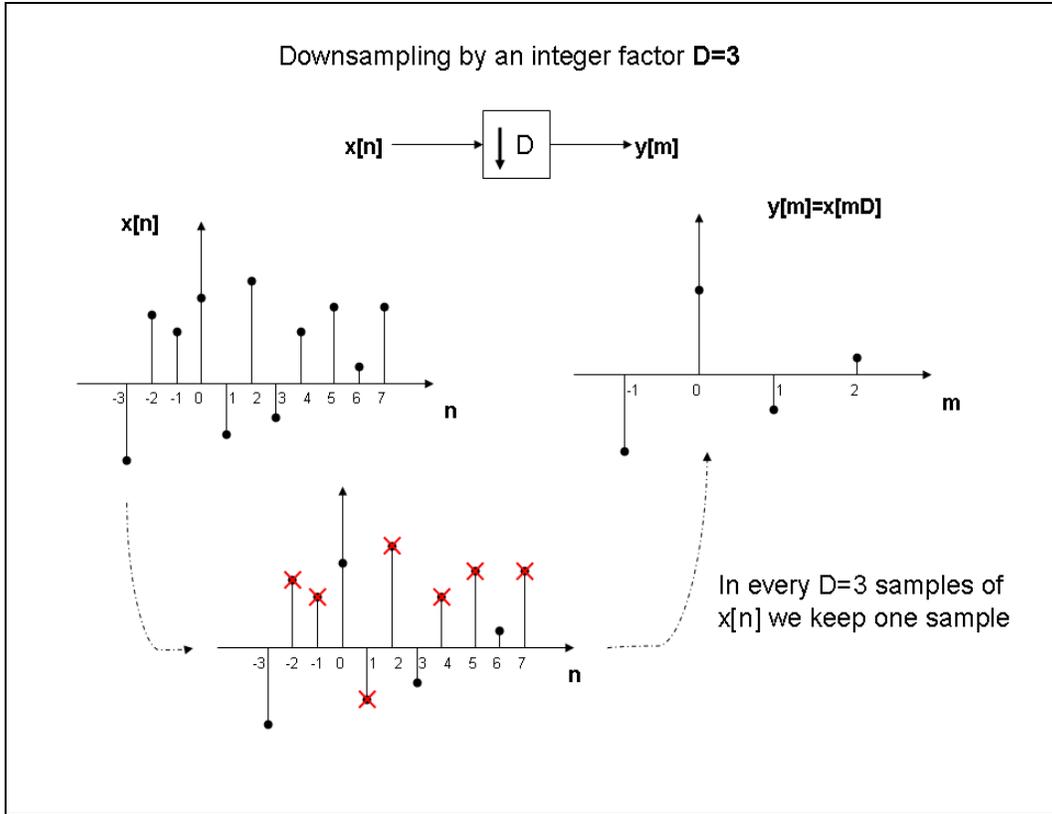


Figure 17. Downsampling Operation.

Consequently, the decimation procedure introduces a loss of information due to the elimination of some data points, so we need to be careful in order to preserve the necessary information of the signal. Distortion of a signal caused by the downsampling operation is in terms of additional frequency components in the frequency spectrum of the resampled signal. This phenomenon is called *aliasing* and it is avoided by properly filtering the signal before downsampling [8].

When a signal sampled at rate  $F_{s_1}$  with frequency spectrum  $X(f_1)$  (in terms of digital frequency) is resampled at a lower sampling rate  $F_{s_2} = \frac{F_{s_1}}{D}$  (where  $D$  is an integer), the resulting frequency spectrum of the resampling signal is given by the following expression [8].

$$Y(f_2) = \frac{1}{N} \sum_{k=1}^{N-1} X\left(\frac{f_2}{N} - \frac{k}{N}\right) \quad (3.2)$$

From equation (3.2) it is easy to show that no aliasing occurs if the signal has no frequencies above  $|f| > \frac{1}{2D}$ , in which case equation (3.2) becomes  $Y(f_2) = \frac{1}{N} X\left(\frac{f_2}{N}\right)$

[9]. Figure 18 illustrates this concept.

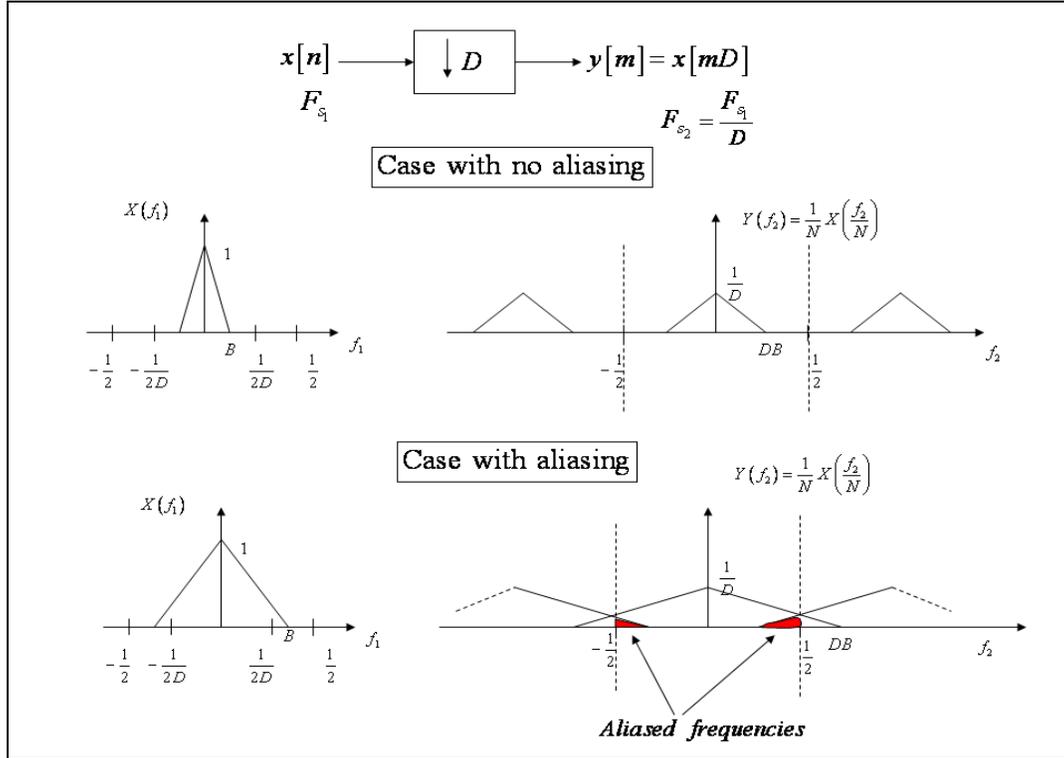


Figure 18. Aliasing Effect in Frequency Spectrum.

Generally, in order to efficiently downsample a noisy signal by an integer factor of  $D$ , with information frequency content within the interval  $\left(-\frac{1}{2D}, \frac{1}{2D}\right)$  and without introducing aliasing, it is necessary to filter the signal first by the appropriate Low Pass Filter (LPF). Therefore, the useful part of the frequency spectrum will be preserved from aliased frequencies caused by noise. Figure 19 illustrates this, along with the specifications of the appropriate Low Pass Filter (LPF).

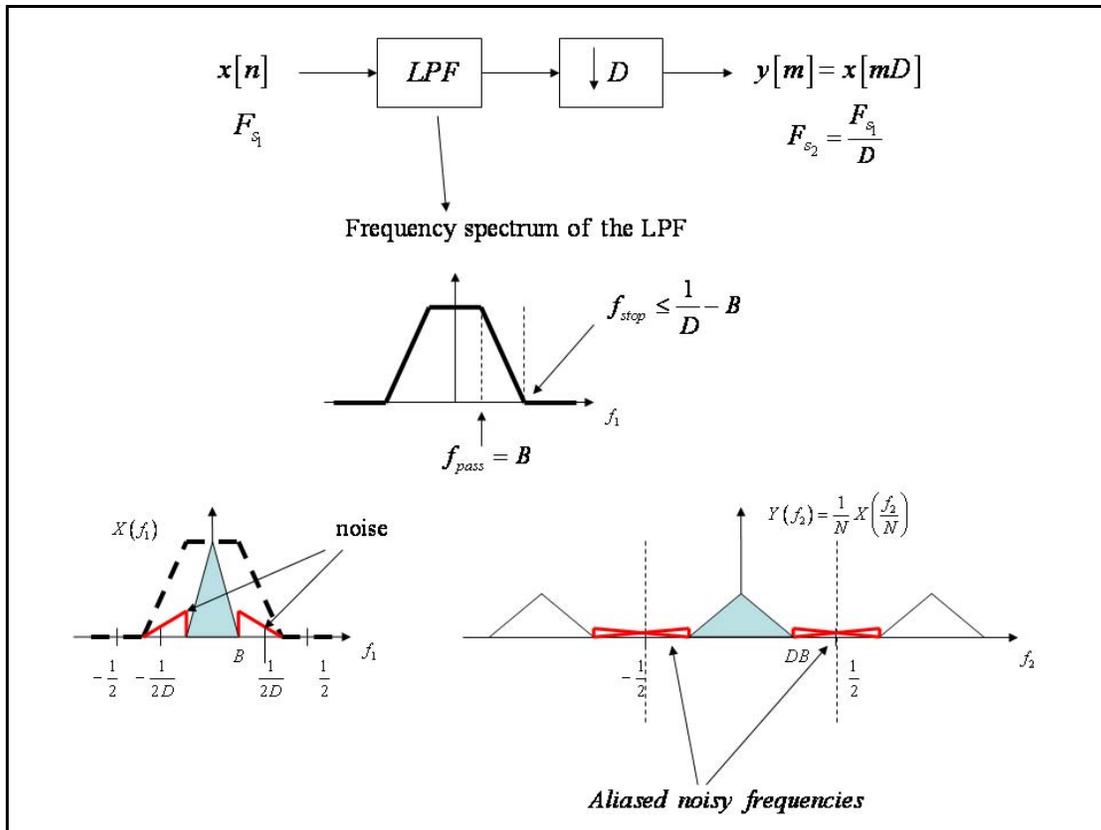


Figure 19. Filtering and Downsampling a Discrete Signal.

### 3. Efficient Implementation of Decimation Operation using Noble Identities and Filter's Polyphase Decomposition

An efficient way of implementing filtering and downsampling operations is by using the Noble identities and the filter's polyphase decomposition. Since the filter in Figure 19 is operated at a higher sampling rate  $F_{s_1}$ , it will be desirable for the filter to be placed after the downsampling operation, resulting in a significant decrease of the number of operations since  $F_{s_2} < F_{s_1}$ . It is well known that by the polyphase decomposition of the filter and the Noble Identities the downsampling operation can be implemented as in Figure 20. In particular, the signal is buffered into  $D$  components at the lower sampling rate and each component is filtered by the polyphased decomposition of the Low Pass Filter [8].

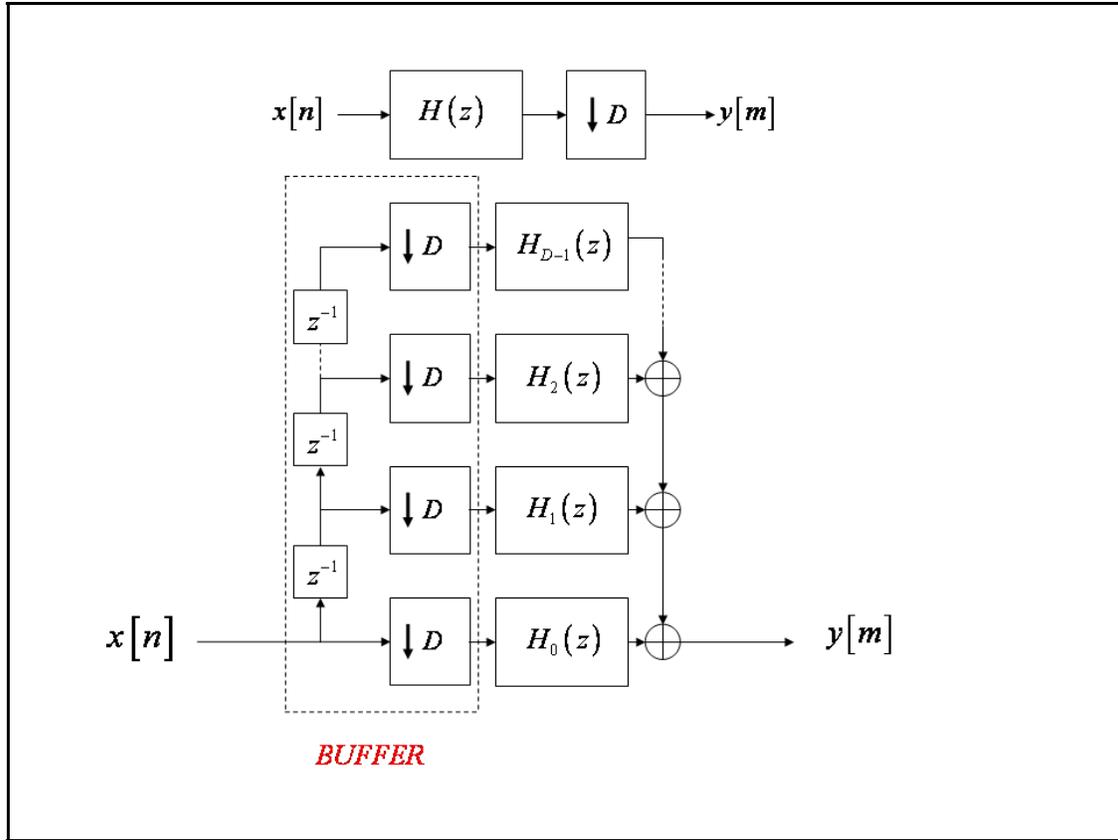


Figure 20. Efficient Implementation of Decimation.

## B. DECIMATION BY TWO WITH FIR MAC AND POLYPHASE DECOMPOSITION

In case of decimation by an integer factor  $D=2$  we can relate the input and output signal as

$$y[n] = \sum_{k=0}^{2N-1} h[k] x[2n-k], \quad (3.3)$$

where  $x[n] = x(nT_s)$ , and  $T_s$  is the sampling interval. Consequently, the output is sampled at half the input rate.

The FIR filter polyphase decomposition provides two components, one for the even samples  $h_0[k] = h[2k]$  and one for the odd samples  $h_1[k] = h[2k+1]$ . Therefore, equation (3.3) can be rewritten as

$$y[n] = \sum_{k=0}^{N-1} h_0[k]x[2(n-k)] + \sum_{k=0}^{N-1} h_1[k]x[2(n-k)-1], \quad (3.4)$$

which breaks down the computation into two phases associated to the even and odd samples, respectively. Equation (3.4) can be rewritten as

$$y[n] = h_0[0]x[2n] + \sum_{k=1}^{N-1} h_0[k]x[2(n-k)] + h_1[0]x[2n-1] + \sum_{k=1}^{N-1} h_1[k]x[2(n-k)-1]. \quad (3.5)$$

Equation (3.5) highlights the fact that, during the time computational interval  $(2n-2)T_s < t \leq (2n)T_s$  the data vector needs to be updated with samples  $x[2n-1]$  and  $x[2n]$ , while the data in the two summations are available before time  $(2n-2)T_s$ .

## 1. Software Implementation

The Simulink/Xilinx implementation needed to perform the decimation-by-two has the same structure as the model presented in Figure 4 with modified parameters to match this case. Specifically, the initial values of the vectors of the Dual Port Ram along with the controller (logic circuit responsible for arranging data points and FIR filter's coefficients) are changed in order to implement equation (3.5). Furthermore, the input data is upsampled at a rate equal to the System Generator's clock rate and the outcome is downsampled twice the Sysgen rate, implementing the decimation-by-two operation. Figure 21 illustrates the structure of this specific design.

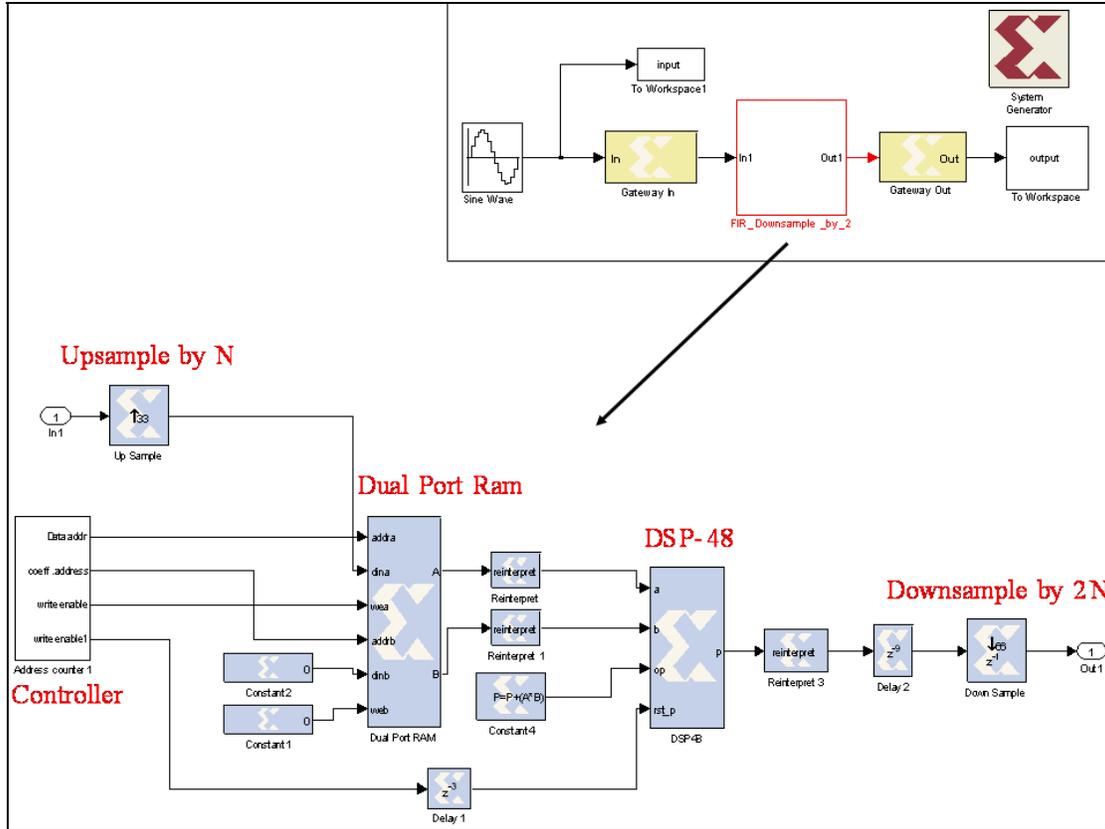


Figure 21. Downsampling by Two.

In order to test the performance of the simulation a sinusoidal signal is provided as an input. The input signal is sampled at rate  $F_s$  while System Generator (Sysgen) works at a higher sampling rate equal to  $NF_s$ , with  $2N-2$  being the degree of the transfer function of the FIR filter which is decomposed into its polyphase components. The generation of the polyphase filter is accomplished in the initialization of the simulation. Since the new system rate provided by Sysgen is higher, the input data is upsampled by the integer factor of  $N$  with the corresponding Xilinx block.

The objective is to achieve a proper alignment of the data and filter's coefficients so that they can be applied to a MAC resulting in the decimation-by-two operation. Towards this goal, we need two memory vectors  $x$  and  $h$ , containing the data and the filter coefficients provided by the Dual Port Ram, and a MAC provided by the DSP48.

*a. Control Logic for Data and Filter Coefficients*

The vector  $h$  of the filter coefficients is defined as  $h = [h[0], h[2], \dots, h[2N-2], h[1], h[3], \dots, h[2N-3], 0]$  and it is the concatenation of the two polyphase components (one for the even and one for the odd samples) of a  $2N-1$  length FIR filter (which is generated in MATLAB) with an additional zero at the end. The vector  $h$  has total length  $2N$  and remains unchanged during the operation of downsampling-by-two. Therefore the ports ‘dinb’ (data input b) and ‘web’ (write enable b) are set to false. The last zero coefficient of vector  $h$  is added in order to serve computational issues derived from the use of DSP48, which works as a MAC and it will be explained in the MAC procedure.

The input data vector stored in the first part of the memory of the Dual Port Ram is a vector  $x$  of length  $2N$  and updated at times  $t = nT_s$  as  $X[(n)_{2N-1}] \leftarrow x[2n]$  for the even samples and  $X[(n-N)_{2N-1}] \leftarrow x[2n-1]$  for the odd samples, with  $(n)_{2N-1} = 0, 1, \dots, 2N-2$  denoting modulo operation. In the implementation,  $(n)_{2N-1}$  is a periodic counter with update rate  $F_{ac} = NF_s$ . The initial value of the memory vector  $x$  is set to the initial conditions (say zero, for example) and updates its value according to the corresponding ‘address’ and ‘write enable’ ports provided from the controller. Figure 22 illustrates the structure of the controller.

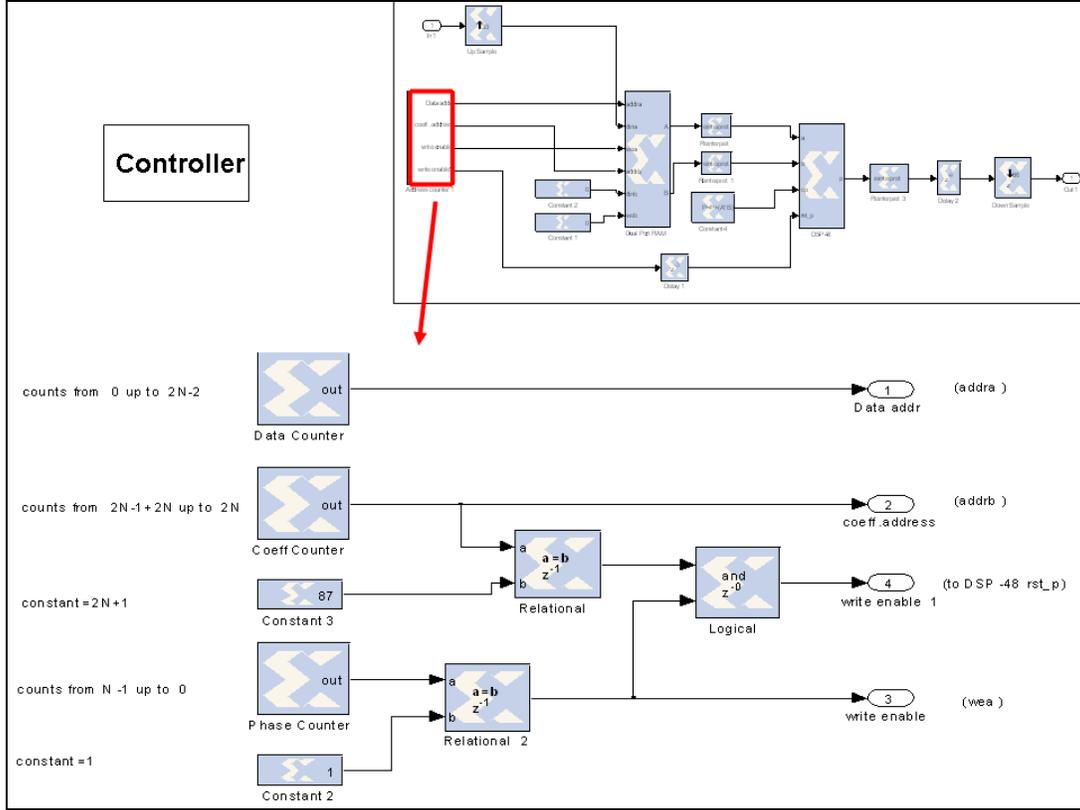


Figure 22. Controller.

The time representation of ‘data address’ and ‘coefficients address’ sequences of Figure 22 is shown in Figure 23. In particular, at time  $(2n-2)T_s$  the accumulator is initialized by  $a((2n-2)T_s) = 0$  (where ‘a’ denotes accumulation function). At every subsequent clock cycle  $T_{ac} = \frac{1}{F_{ac}}$  the accumulation will be updated by

$$a((2n-2)T_s + \lambda T_{ac}) = a((2n-2)T_s + (\lambda-1)T_{ac}) + (data\_addr(\lambda) \cdot coeff\_addr(\lambda)),$$

where  $\lambda = 1, \dots, 2N$ . At time  $(2n-1)T_s = (2n-2)T_s + NT_{ac}$  the input data is updated. The output  $y[n]$  at time  $2nT_s$  is shown in the timing diagram of Figure 23.

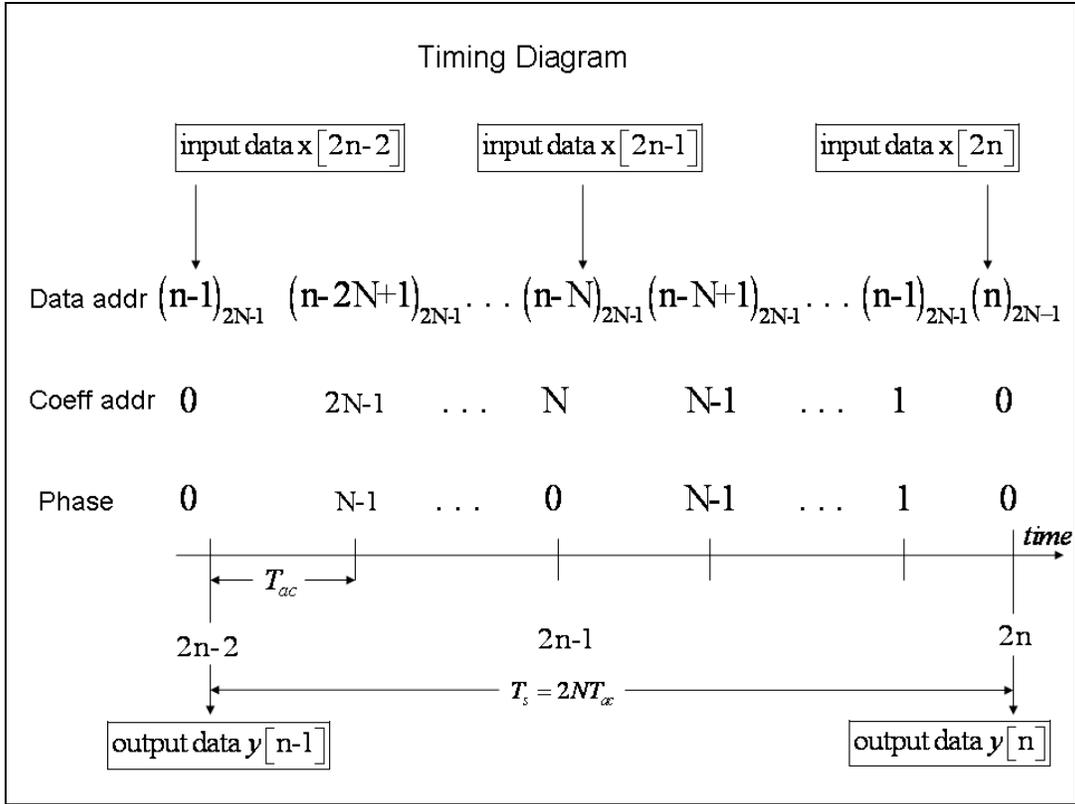


Figure 23. Time Representation of Simulation.

In order to demonstrate the functionality of the implementation, Figure 23 illustrates the timing of the various signals involved.

The outcome of the Dual Port Ram is a set of bitstreams, one from port  $A$  (data points) and one from port  $B$  (filter coefficients). It can be inferred that the outcome of port  $A$  is a recurrent window of length  $2N$ , which is subdivided into two windows (one for the even samples and one for the odd samples of input signal) of length  $N$ . At every time  $T_{ac}$  both the even and the odd samples are updated, introducing a shift by one position from left to right. The bitstream of port  $B$  is a repetition of the vector  $h$ . Figure 24 illustrates the outcome of Dual Port Ram.

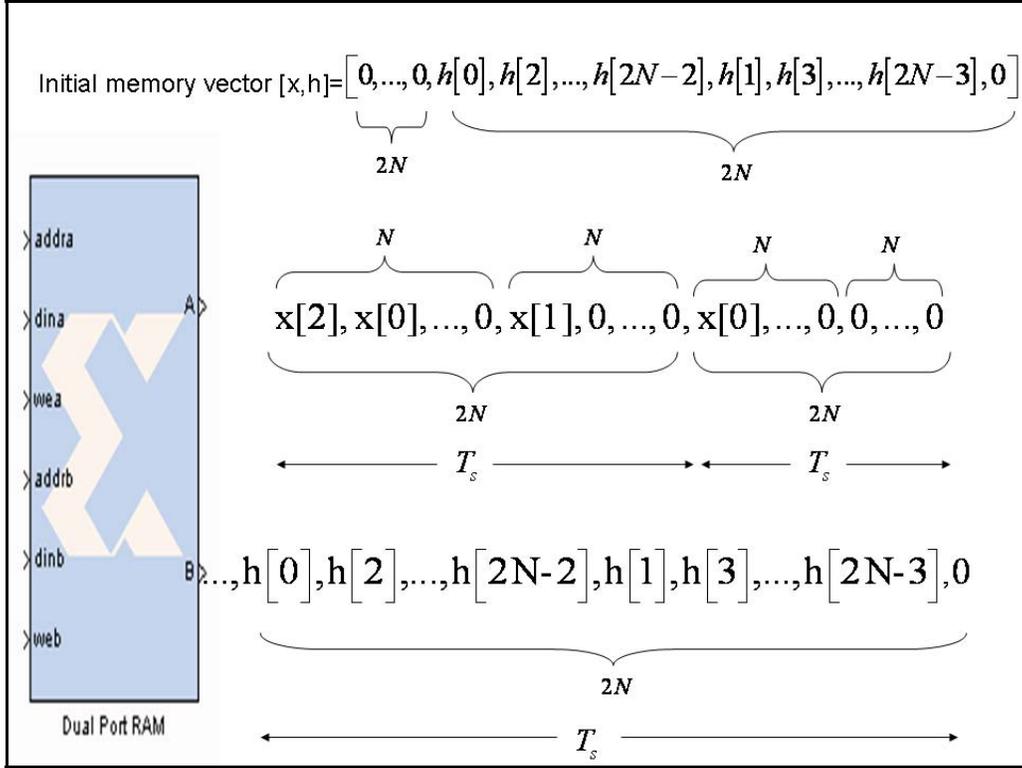


Figure 24. Outcome of Dual Port Ram.

**b. Sequential Multiplication and Accumulation (MAC) of Data and Filter Coefficients using the DSP48**

The output bitstream from the Dual Port Ram as it is shown in Figure 24 is being processed by the DSP48 Xilinx block, which works as a MAC. Its operation mode is defined by  $P = P + A \cdot B$  (referring to Figure 7) where the product of two output pairs  $A$  and  $B$  of the Dual Port Ram, is being accumulated each time with the previous product. A reset signal (selected from the DSP48 options) for the outcome  $P$  is introduced at clock rate  $2N$  provided from the properly delayed 'write enable 1' signal of the controller of the Dual Port Ram (referring to Figure 22). The adjustment of the delay is set so that the reset of the outcome  $P$  occurs every  $2N$  times, where a data coefficient is multiplied with the zero coefficient of vector  $h$ .

Consequently, considering the length ( $2N$ ) of block pairs in figure 24, after the last product  $x[k] \cdot h[0]$  is accumulated to the previous  $2N$  sums of products of

each block pair, a data point of decimation-by-two operation  $y[n]$  is generated as shown also in the timing diagram (Figure 23). Figure 25 shows the first two points  $y[0]$ ,  $y[1]$  computed at times  $(2N)T_{ac}$  and  $(4N)T_{ac}$ .

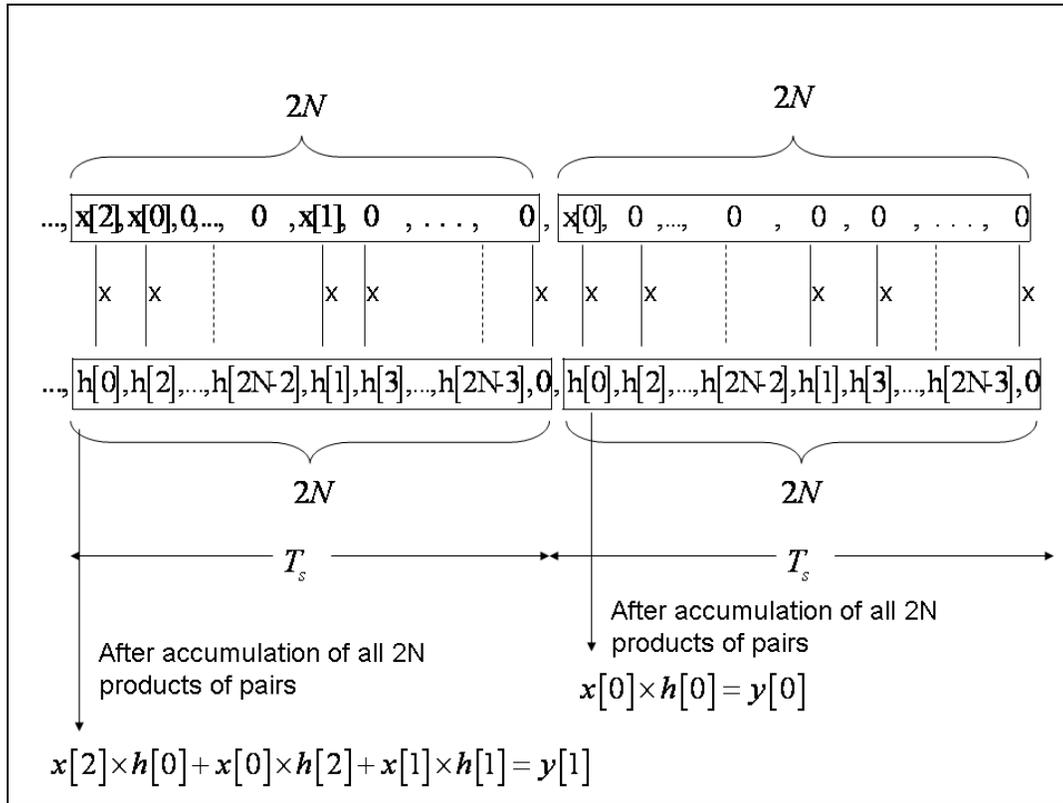


Figure 25. Outcome of DSP48.

Referring to Figure 7, the bitstream outcome  $P$  of the DSP48 can be considered as a set of blocks of length  $2N$  in which the desired coefficients of the decimation-by-two operation are embedded in every  $(2N)^{th}$  element of each block as shown in the timing diagram in Figure 23. Therefore, by downsampling the data  $P$  by the factor of  $2N$  (twice the factor that was used when the input data was upsampled) the desired decimation-by-two operation is performed.

*c. Results*

In order to test the performance of the simulation a sinusoidal waveform with frequency  $F = 0.2 \cdot F_s$  (Hz) and sampling frequency  $F_s = 1$  (Hz) is applied as an input.

The FIR filter has been designed as an Equiripple filter and decomposed into two polyphase components with the following characteristics:

Passband: 0-0.3 (in terms of Digital Frequency  $f$ )

Stopband: 0.4-0.5 (in terms of Digital Frequency  $f$ )

Order: 65

The frequency spectrum of the original and the downsampled-by-two signal is shown in Figure 26. We can verify that the frequency spectrum of the downsampled-by-two signal is stretched (in terms of the digital frequency) by the integer factor of two compared to the frequency spectrum of the original signal. Since the bandwidth of the signal is less than  $\frac{1}{4}$  there is no aliasing effect. Therefore, the frequency of the original signal is  $f = 0.2$  while the frequency of the downsampled-by-two signal is  $f = 2 \times 0.2 = 0.4$  (where  $f$  is the dimensionless digital frequency).

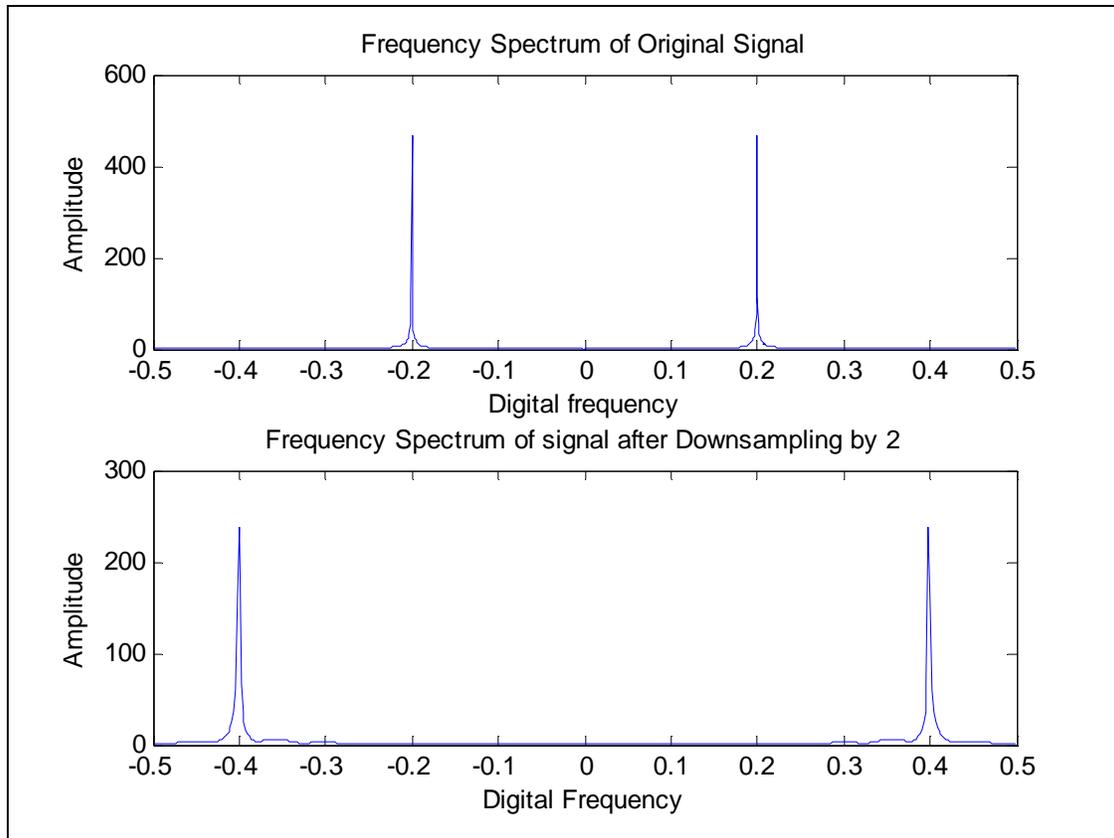


Figure 26. Frequency Spectrum of the Original and Downsampled by Two Signal without Aliasing Effect.

In order to demonstrate the aliasing effect in the frequency spectrum of a downsampled signal, a sinusoidal waveform with frequency  $F = 0.3 \cdot F_s$  (Hz) is applied as an input. Since the new bandwidth (0.3) exceeds the factor  $\frac{1}{2D}$  (where  $D = 2$ ), the new frequency spectrum of the downsampled-by-two signal in the interval  $-\frac{1}{2} \leq f \leq \frac{1}{2}$  will contain aliased frequencies derived from the periodic repetition of one period of the frequency spectrum. Figure 27 illustrates this example.

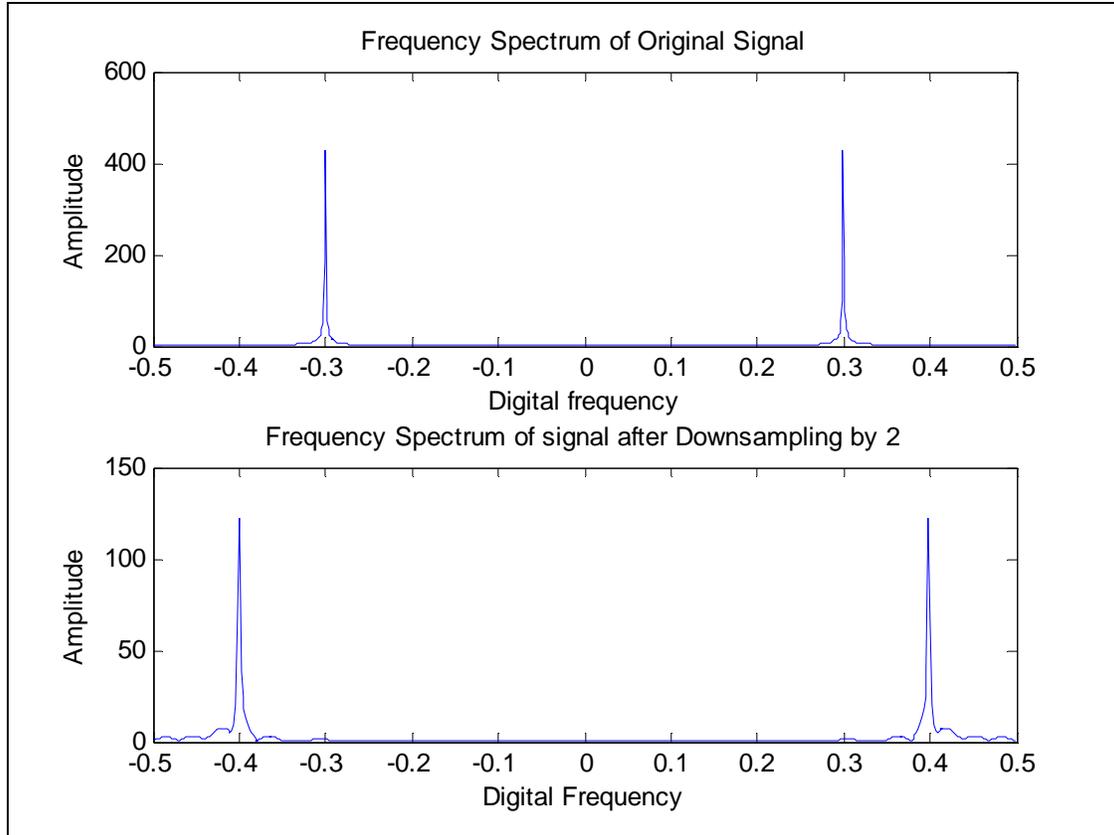


Figure 27. Frequency Spectrum of the Original and Downsampled by Two Signal with Aliasing Effect.

### C. DECIMATION BY AN INTEGER FACTOR ‘D’ WITH FIR MAC AND POLYPHASE DECOMPOSITION

The structure of the decimation-by-two operation can be easily extended to a more general decimation-by- $D$  operation for any  $D \geq 2$ . The decimated signal obtained from an input signal  $x[n]$ , which is filtered by a FIR filter  $h[n]$  (decomposed into its polyphase components) and then downsampled by an integer factor of  $D$  is given by

$$y[n] = \sum_{k=0}^{DN-1} h[k]x[nD-k], \quad (3.6)$$

with  $D$  integer and  $x[n] = x(nT_s)$ ,  $y[n] = y(nDT_s)$  the input and output sequences sampled at rates  $F_s = 1/T_s$  and  $F_s/D = 1/(DT_s)$  respectively.

The  $D$  polyphase components of the FIR filter are defined by

$$h_\ell[k] = h[kD + \ell], \quad (3.7)$$

with  $\ell = 0, \dots, D-1$  and  $k = 0, \dots, N-1$ . The decimated output is the superposition of the  $D$  phases and it is given by the following expression:

$$\begin{aligned} y[n] = & \sum_{k=0}^{N-1} h_0[k]x[(n-k)D] + \sum_{k=0}^{N-1} h_1[k]x[(n-k)D-1] + \dots \\ & \dots + \sum_{k=0}^{N-1} h_{D-1}[k]x[(n-k)D-D+1]. \end{aligned} \quad (3.8)$$

Equation (3.8) can be further decomposed as:

$$y[n] = \dots + h_\ell[0]x[nD - \ell] + \sum_{k=1}^{N-1} h_\ell[k]x[(n-k)D - \ell] + \dots \quad (3.9)$$

During the time computational interval  $(Dn - D)T_s < t \leq (Dn)T_s$  the data vector needs to be updated with samples  $x[Dn - (D-1)]$  up to  $x[Dn]$ , while the data in the  $D$  summations are available before time  $(Dn - D)T_s$ .

The design needed to perform the decimation-by- $D$  operation is similar to the decimation-by-two case. The memory vector for the input data points in the Dual Port Ram has length  $DN - 1$  and updates its value by

$$\begin{aligned} X[(n)_{DN-1}] & \leftarrow x[nD], \\ X[(n-N)_{DN-1}] & \leftarrow x[nD-1], \\ \dots & \\ X[(n-\ell N)_{DN-1}] & \leftarrow x[nD-\ell], \\ \dots & \\ X[(n-(D-1)N)_{DN-1}] & \leftarrow x[nD-D+1]. \end{aligned}$$

The FIR filter coefficients vector, which is stored in the second memory of the Dual Port Ram is the concatenation of its polyphase components derived from expression (3.7) with total length  $DN - 1$ .

In the implementation,  $(n)_{DN-1} = 0, \dots, DN-2$  is a periodic counter with update rate  $F_{ac} = NF_s$ , which is the clock rate of the System Generator. Therefore the input data is upsampled by the integer factor of  $N$ .

The time representation of ‘data address’ and ‘coefficients address’ sequences are shown in Figure 28. In particular, at time  $(Dn-D)T_s$  the accumulator is initialized as  $a((Dn-D)T_s) = 0$  (where ‘ $a$ ’ denotes the accumulation function). At every subsequent clock cycle  $T_{ac} = \frac{1}{F_{ac}}$  the accumulation will be updated by

$$a((Dn-D)T_s + \lambda T_{ac}) = a((Dn-D)T_s + (\lambda-1)T_{ac}) + (data\_addr(\lambda) \cdot coeff\_addr(\lambda)),$$

where  $\lambda = 1, \dots, DN$ . The input data is updated every  $N^{th}$  multiple of  $T_{ac}$  with total multiples  $DN$ . In particular,  $(Dn-D+1)T_s = (Dn-D)T_s + NT_{ac}$ . The output  $y[n]$  at time  $DnT_s$  is shown in the timing diagram of Figure 28.

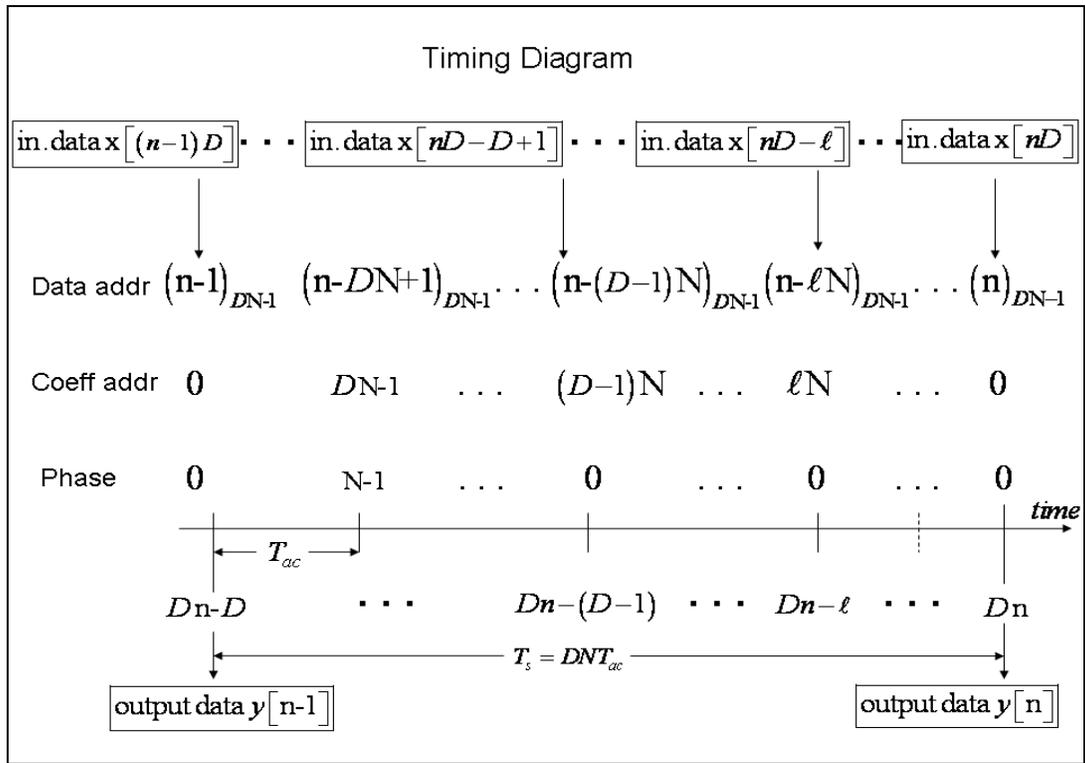


Figure 28. Timing Diagram for Decimation by D.

Apart from the new vectors that are stored in the Dual Port Ram, this design can be obtained by simple extension of the decimation-by-two case to the more general decimation-by-D.

Referring to Figure 7, the bitstream outcome  $P$  of the DSP48 can be considered as a set of blocks of length  $DN$  in which the desired coefficients of the decimation-by- $D$  operation are embedded in every  $(DN)^{th}$  element of each block as shown in the timing diagram in Figure 28. Therefore by downsampling the data  $P$  by the factor of  $DN$  ( $D$  times the factor which was used when the input data was upsampled) the desired decimation-by-D operation is performed.

In order to test the performance of the simulation for the decimation factor  $D = 4$  a sinusoidal waveform with frequency  $F = 0.1 \cdot F_s$  (Hz) and sampling frequency  $F_s = 1$  (Hz) is applied as an input.

The FIR filter has been designed as an Equiripple filter and decomposed into four polyphase components with the following characteristics:

Passband: 0-0.2 (in terms of Digital Frequency  $f$ )

Stopband: 0.25-0.5 (in terms of Digital Frequency  $f$ )

Order: 29

The frequency spectrum of the original and the downsampled by  $D = 4$  signal is shown in Figure 29. We can verify that the frequency spectrum of the downsampled signal is stretched (in terms of the digital frequency) by the integer factor of four compared to the frequency spectrum of the original signal. Since the initial bandwidth of the signal is less than  $\frac{1}{2D}$  there is no aliasing effect. Therefore, the frequency of the original signal is  $f = 0.1$ , while the frequency of the decimation-by-four signal is  $f = 4 \times 0.1 = 0.4$ , where  $f$  is the dimensionless digital frequency.

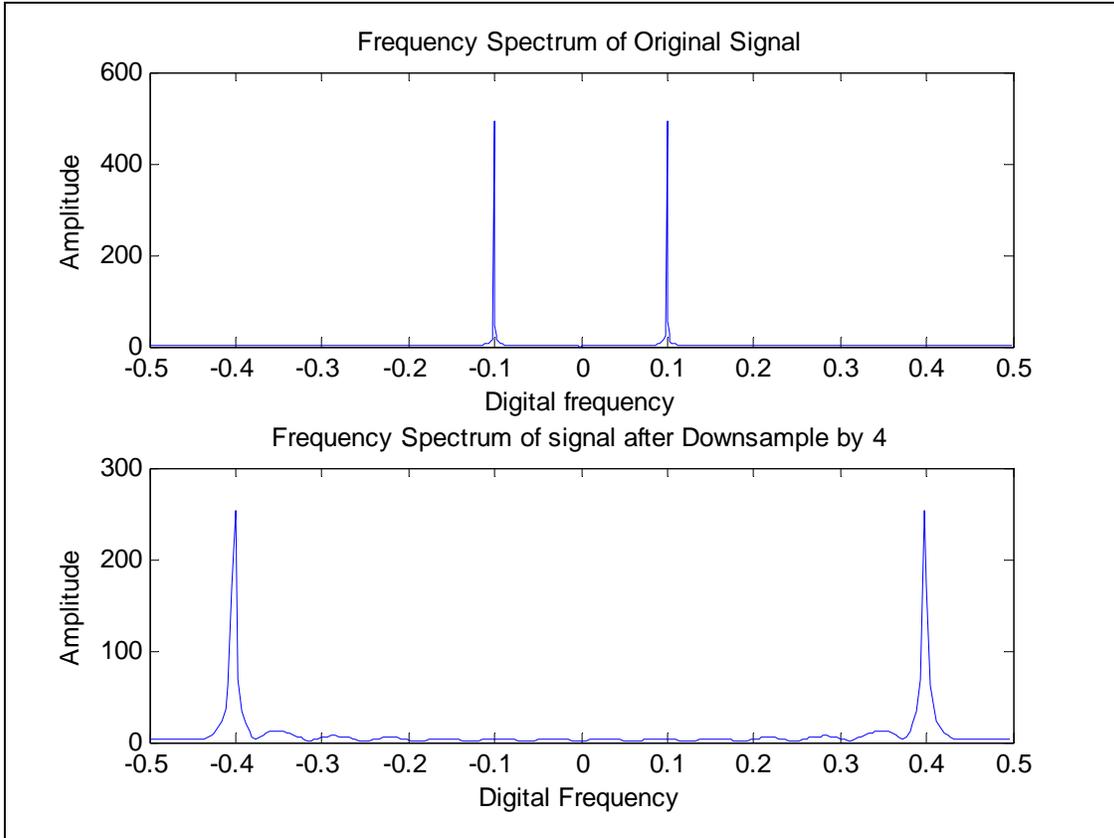


Figure 29. Frequency Spectrum of the Original and Downsampled by  $D = 4$  Signal.

## IV. INTERPOLATION BY AN INTEGER FACTOR

### A. THEORETICAL PERSPECTIVE

#### 1. Analysis of Upsampling (Interpolation)

In Software Defined Radios (SDR), the modulation process is performed in the digital domain. The data rate of the transmitted information needs to be increased in order to match the rate of the modulation (carrier frequency). An upsample operation (interpolation) increases the number of samples per second of a given signal by an integer factor  $D$ . An example of interpolation by integer factor of  $D = 3$  is shown in Figure 30.

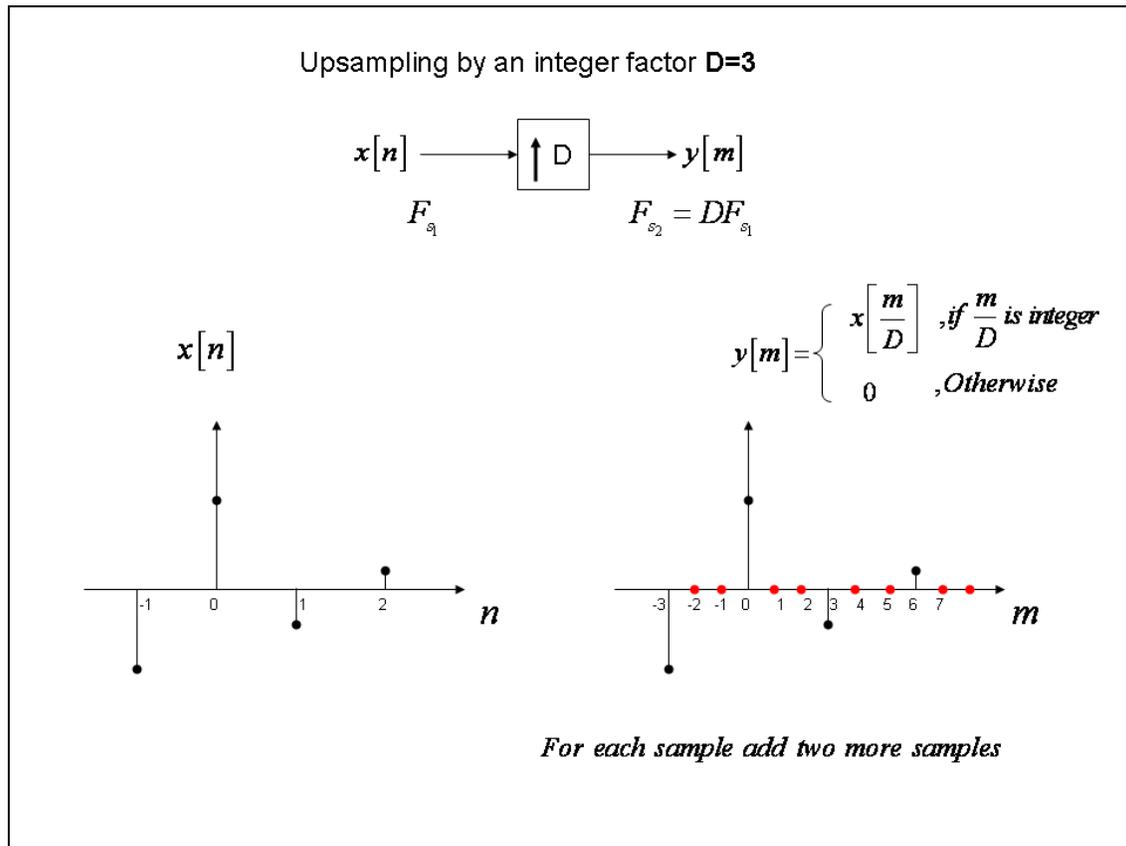


Figure 30. Upsampling Operation.

When a signal sampled at a rate  $F_{s_1}$  with frequency spectrum  $X(f_1)$  (in terms of digital frequency) is resampled at a higher rate  $F_{s_2} = DF_{s_1}$ , where  $D$  is an integer, the resulting frequency spectrum of the resampled signal is given by the following expression:

$$Y(f_2) = X(f_1) \Big|_{f_1 = Df_2} \quad (4.1)$$

It is obvious from equation (4.1) that the new frequency spectrum is ‘squeezed’ in terms of the digital frequency (horizontal axis) [8]. Consequently, since the frequency spectrum of the resampled signal is a periodic repetition of one period between the interval  $\left(-\frac{1}{2}, \frac{1}{2}\right)$ , additional *image frequency* components (‘ghost’ frequencies) will appear in the spectrum of the upsampled signal. These frequencies are artifacts created by the upsampling operation. The frequency spectra of the original signal and after upsampling by  $D$  is shown in Figure 31.

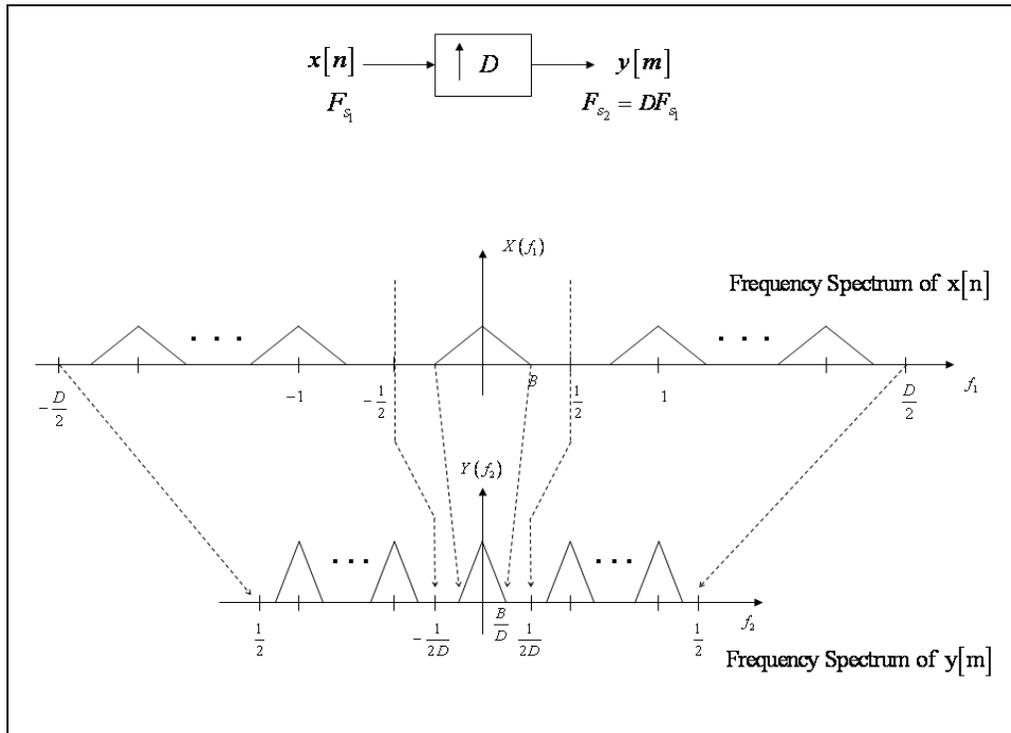


Figure 31. ‘Ghost’ Frequencies in Interpolation Operation.

In order to eliminate the ‘ghost’ frequencies a Low Pass Filter (LPF) is needed after the upsampling operation. The frequency response of the LPF along with its specifications is depicted in Figure 32.

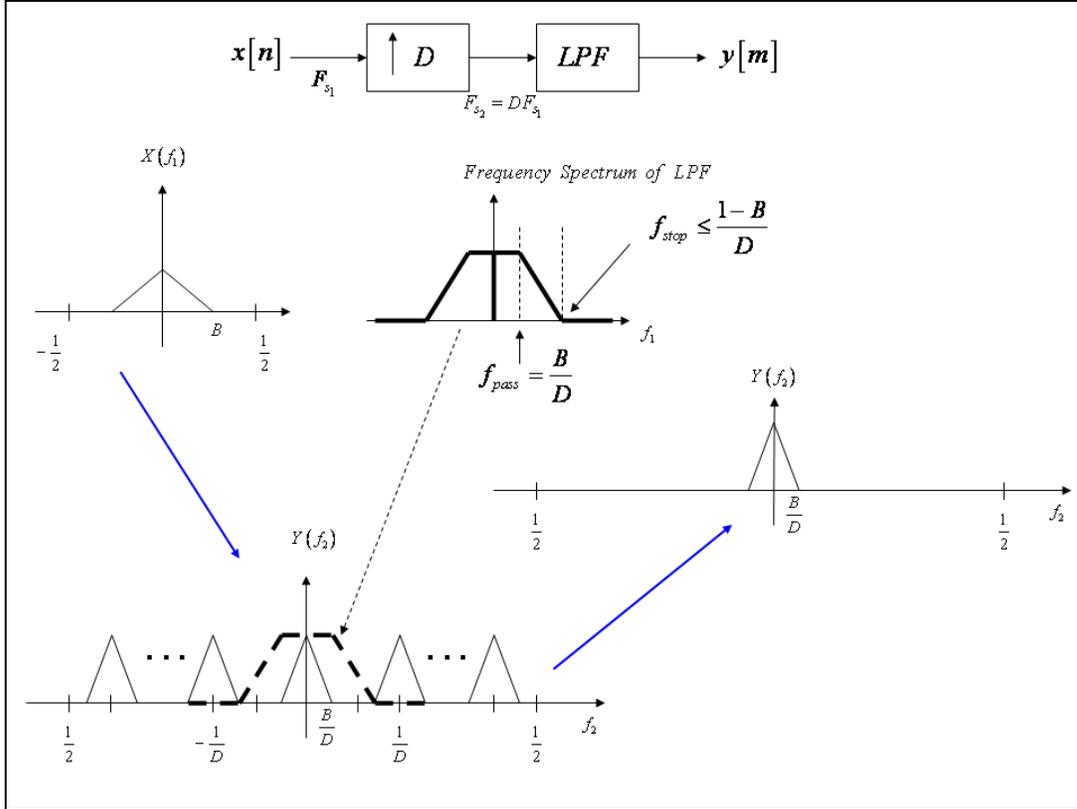


Figure 32. Upsampling and Filtering with LPF.

## 2. Efficient Implementation of Interpolation Operation using Noble Identities and Filter’s Polyphase Decomposition

An efficient way of implementing upsampling and filtering operations is by using the Noble identities with the filter’s polyphase decomposition. Since the filter in Figure 32 is operated at a higher sampling rate  $F_{s_2}$  it would be desirable for the filter to be placed before the upsampling operation, thus minimizing the cost. It can be shown that the upsampling operation shown in Figure 32, with the LPF  $H(z) = \sum_{n=0}^N h(n)z^{-n}$ , can be

implemented as shown in Figure 33, where the filters  $H_k(z)$ , for  $k = 0, \dots, D-1$  are the polyphase components of  $H_k(z) = \sum_{n=0}^{N/D} h(nD+k)z^{-n}$  [8].

The upsampling network on the right of Figure 33, after the filters, is an ‘interlacer’ that interlaces the outputs of all  $D$  filters, thus increasing the sampling rate. This implementation is particularly attractive, since it has the same complexity as the original but is implemented at the lowest sampling rate [8].

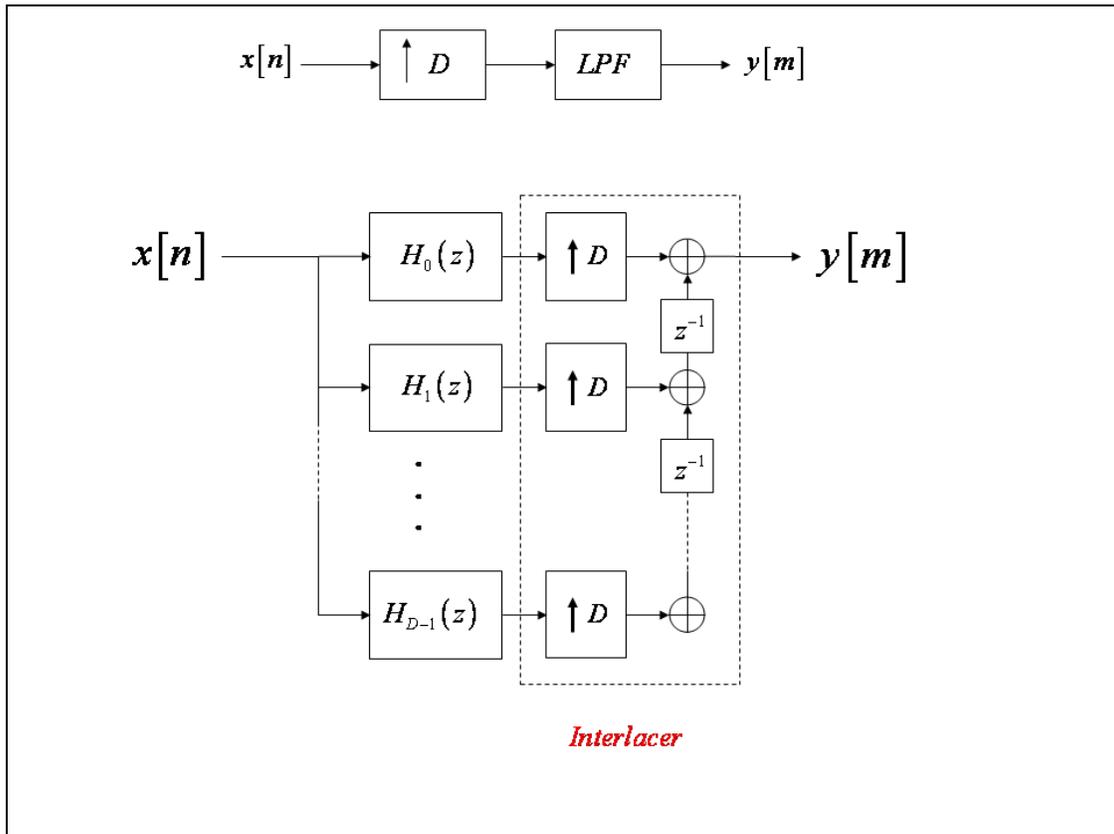


Figure 33. Efficient Implementation of Interpolation.

## B. INTERPOLATION BY TWO WITH FIR MAC AND POLYPHASE DECOMPOSITION

From the polyphase decomposition the upsampling by two is determined as

$$\begin{aligned}y[2n] &= \sum_{k=0}^{N-1} h_0[k]x[n-k], \\y[2n+1] &= \sum_{k=0}^{N-1} h_1[k]x[n-k].\end{aligned}\tag{4.2}$$

Here  $h_0[k] = h[2k]$  and  $h_1[k] = h[2k+1]$  are the polyphase components (even and odd samples) of the filter  $h[n]$  while  $x[n] = x(nT_s)$  with  $T_s$  the sampling interval. Consequently, the output rate is twice the input rate.

Equation (4.2) highlights the fact that the signal  $x[n]$  is interpolated by interlacing two signals,  $y[2n]$  and  $y[2n+1]$ , which are computed independently.

### 1. Software Implementation

The Simulink/Xilinx implementation needed to perform the interpolation-by-two has the same structure as the model presented in Figure 4 with parameters properly chosen to match the new case. Specifically the initial values of the vectors of the Dual Port Ram along with the controller (logic circuit responsible for arranging data points and FIR filter's coefficients) are changed in order to implement equation (4.2). Furthermore, the input data is upsampled at a rate equal to the System Generator's clock rate and the outcome is downsampled at the half of the Sysgen rate, implementing the interpolation-by-two operation. Figure 34 illustrates this design.

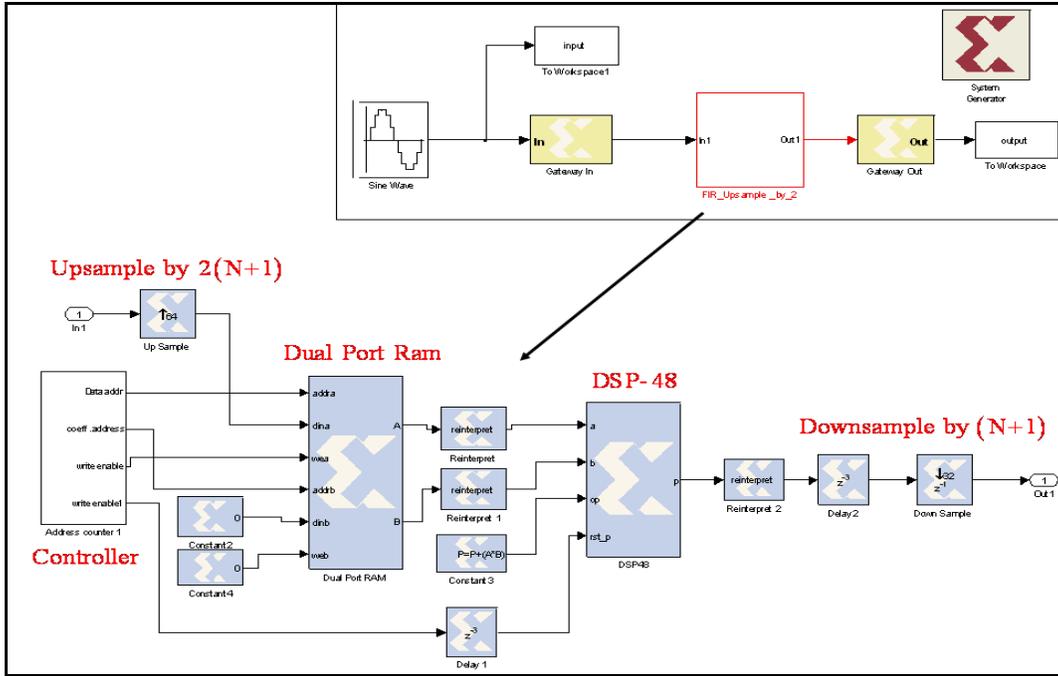


Figure 34. Upsampling by Two.

In order to test the performance of the simulation, a sinusoidal signal is provided as an input. The input signal is sampled at rate  $F_s$ , while the System Generator (Sysgen) works at a higher sampling rate equal to  $2(N+1)F_s$ , with  $2N-2$  being the degree of the transfer function of the FIR filter which is decomposed into its polyphase components. The generation of the polyphase filter is accomplished in the initialization of the simulation. Since the new system rate provided by Sysgen is higher, the input data is upsampled by the integer factor of  $2(N+1)$  with the corresponding Xilinx block.

The objective is to achieve a proper alignment of the data and filter's coefficients so that they can be applied to a MAC resulting in the interpolation-by-two operation. Towards this goal, we need two memory vectors  $x$  and  $h$  containing the data and the filter coefficients respectively provided by the Dual Port Ram and a MAC provided by the DSP48.

*a. Control Logic for Data and Filter Coefficients*

The vector  $h$  of the filter coefficients is defined as  $h = [h[0], h[2], \dots, h[2N-2], 0, 0, h[1], h[3], \dots, h[2N-3], 0]$  and it is the concatenation of the two polyphase components (one for the even and one for the odd samples) of a  $2N-1$  length FIR filter (which is generated in MATLAB) with three properly placed additional zeros. The length of the vector  $h$  is  $2N+2$  and its value remains unchanged during the operation of upsampling-by-two. Therefore the ports ‘dinb’ (data input b) and ‘web’ (write enable b) are set to false. The zero coefficients are required by computational issues derived from the controller and the use of the DSP48 block which works as a MAC and it will be explained in the MAC procedure.

The input data vector stored in the first part of the memory of the Dual Port Ram is a vector  $x$  of length  $N$  and it is updated as  $X[(n)_N] \leftarrow x[n]$ . The data address counter is defined as  $(n)_N = 0, 1, \dots, N-1$  and it is repeated twice during the sampling interval  $T_s$ . The initial value of the memory vector  $x$  is set to the initial conditions (say zero, for example) and updates its value according to the corresponding ‘address’ and ‘write enable’ ports provided from the controller. Figure 35 illustrates the controller of the simulation.

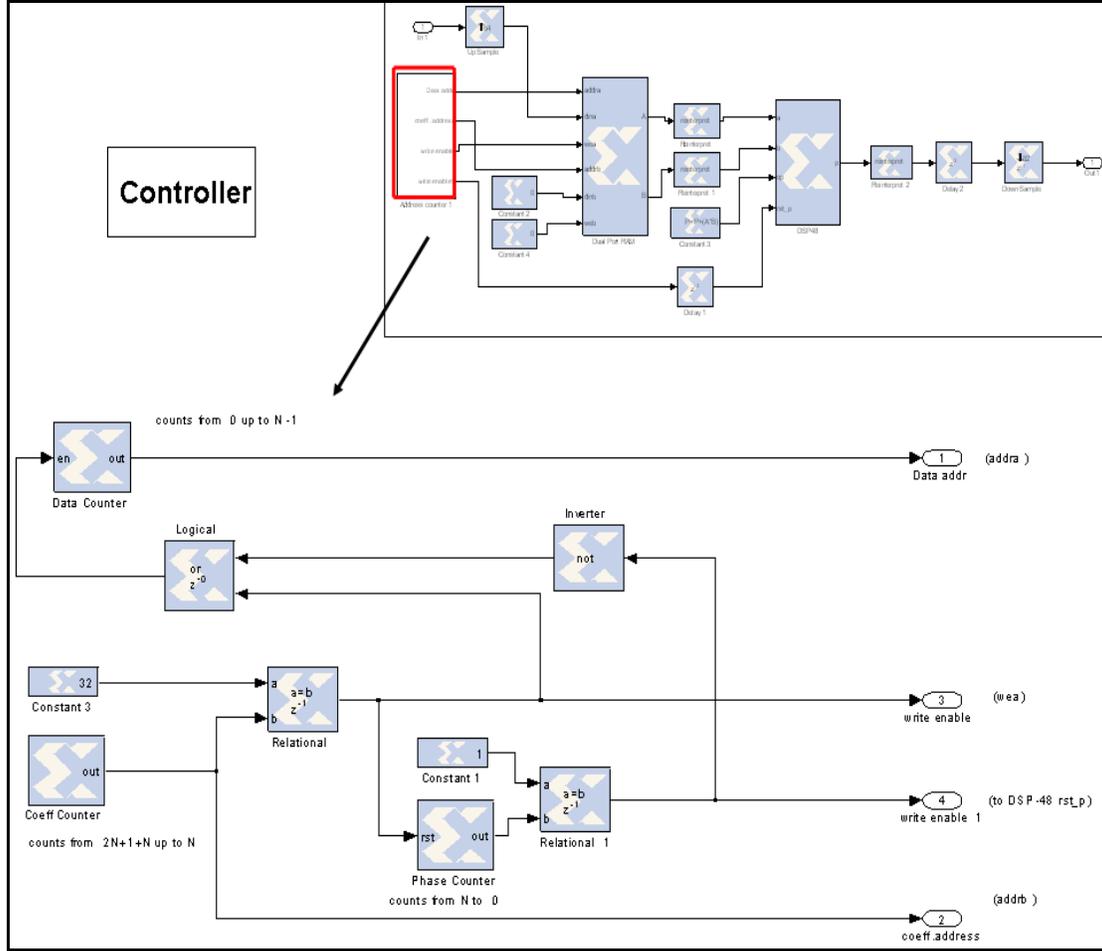


Figure 35. Controller.

The time representation of ‘data address’ and ‘coefficients address’ sequences of Figure 35 are shown in Figure 36. In particular, at time  $(2n-2)T_s$  the accumulator is initialized as  $a((2n-2)T_s)=0$  (where  $a$  denotes the content of the accumulation). At every subsequent clock cycle  $T_{ac} = \frac{1}{F_{ac}}$ , the accumulation will be updated by

$$a((2n-2)T_s + \lambda T_{ac}) = a((2n-2)T_s + (\lambda-1)T_{ac}) + (data\_addr(\lambda) \cdot coeff\_addr(\lambda)), \text{ and}$$

$$a((2n-1)T_s + \lambda T_{ac}) = a((2n-1)T_s + (\lambda-1)T_{ac}) + (data\_addr(\lambda) \cdot coeff\_addr(\lambda)),$$

with  $\lambda = 1, \dots, N$ . The outputs  $y[2n-1], y[2n]$  at times  $(2n-1)T_s$  and  $2nT_s$  respectively are shown in the timing diagram of Figure 36. .

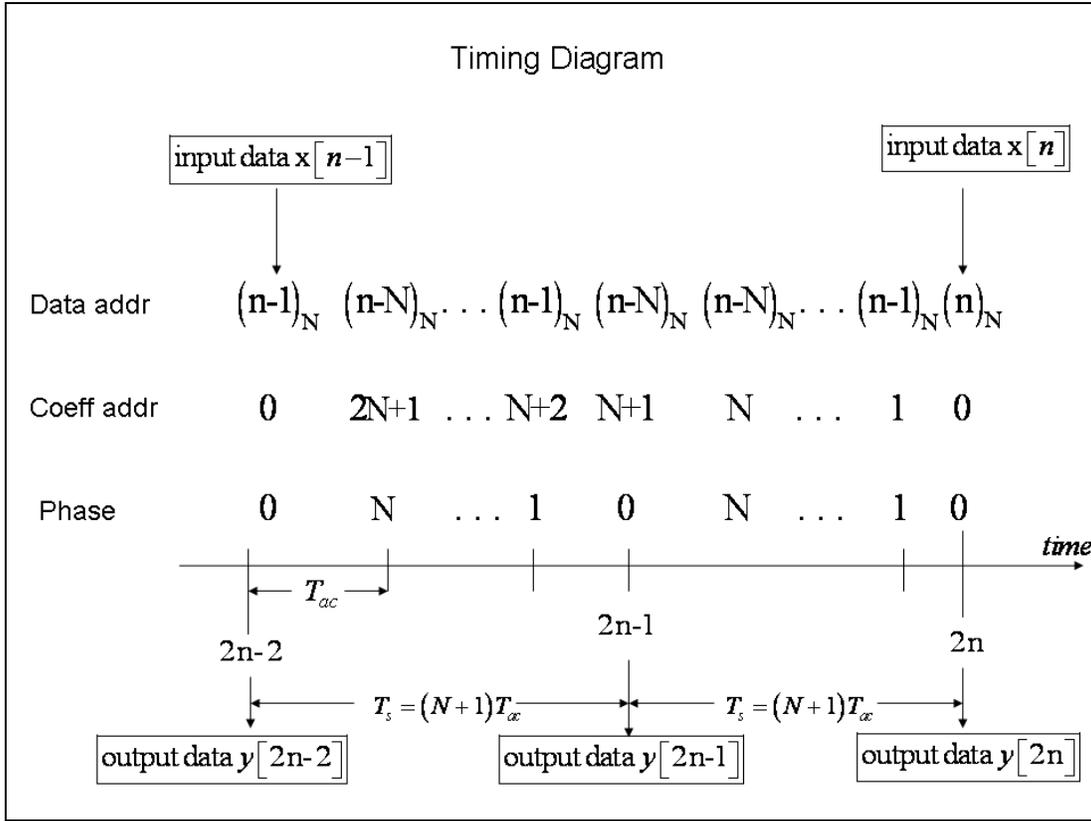


Figure 36. Time Representation of Simulation.

In order to demonstrate the functionality of the implementation, Figure 36 illustrates the timing of the various signals involved.

The outcome of the Dual Port Ram is a set of bitstreams, one from port  $A$  (data points) and one for port  $B$  (filter's coefficients). The bitstream of port  $B$  is a repetition of the vector  $h$ . It can be inferred that the outcome of port  $A$  is a recurrent window of length  $2N+2$ , which is subdivided into two windows of length  $N+1$ . At every time  $T_{ac}$  both subwindows are updated with the same data, introducing a shift by one position from left to right, while the first subwindow starts updating from the second sample. Figure 37 illustrates the outcome of the Dual Port Ram.

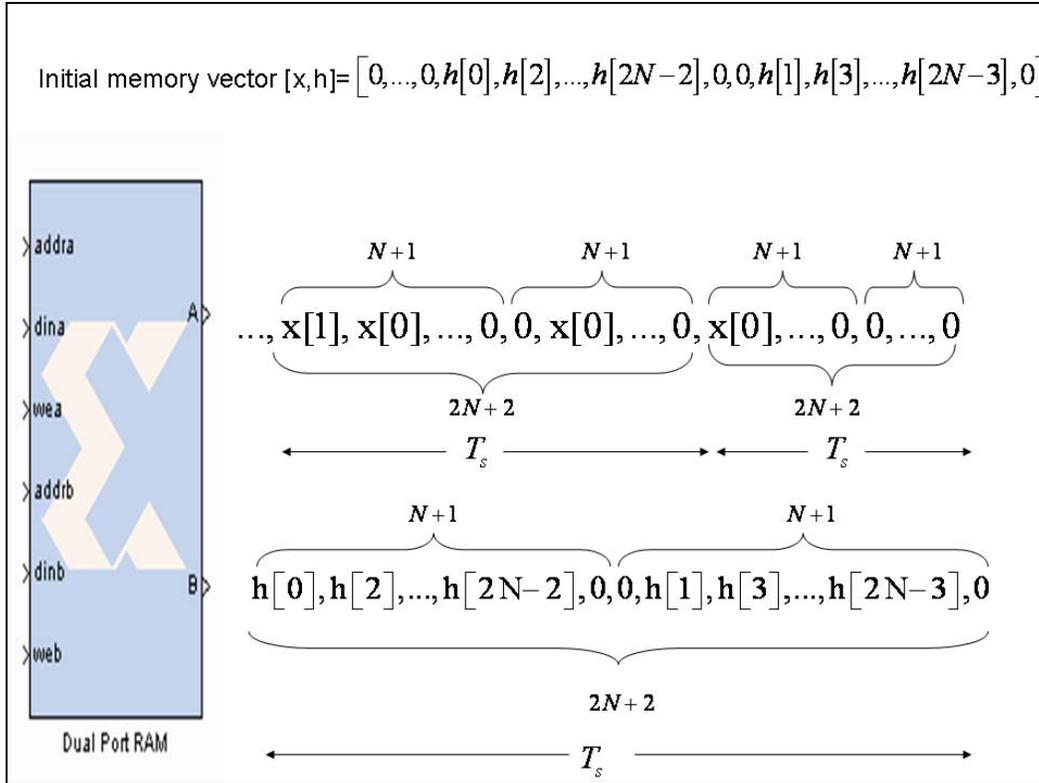


Figure 37. Outcome of Dual Port Ram.

**b. Sequential Multiplication and Accumulation (MAC) of Data and Filter Coefficients using DSP48**

The output bitstream from the Dual Port Ram, as it is shown in Figure 37, is being processed by the DSP48 Xilinx block, which implements the MAC. Its operation mode is defined by  $P = P + A \cdot B$  (referring to Figure 7), where the product of two pairs of Dual Port Ram output ports  $A$  and  $B$  is being accumulated each time with the previous product. A reset signal (selected from the DSP48 options) for the outcome  $P$  is introduced at clock rate  $N + 1$  provided from the properly delayed 'write enable 1' signal of the controller of the Dual Port Ram (referring to Figure 35). The adjustment of the delay is set so that the reset of the outcome  $P$  occurs every  $N + 1$  samples, where the coefficients of vector  $h$  are zero, without affecting the accumulation procedure of the interpolation-by-two operation and therefore there is no loss of information. The third

zero coefficient of vector  $h$  which is placed at the first odd sample is not affecting the accumulation process of the interpolation-by-two operation as well, since the first subwindow of the data vector is updated from the second sample.

Consequently, considering the length  $(2N+2)$  of block pairs in Figure 37, whenever a product  $x[k] \cdot h[0]$  is accumulated to the previous  $2N+1$  sums of products of each block pair, two data points of interpolation-by-two operation  $y[n]$  are provided at every  $(N+1)T_{ac}$  interval as it also shown in the timing diagram in Figure 36. Figure 38 shows the first three points  $y[0], y[1], y[2]$  provided at time  $\mu(N+1)T_{ac}$ , where  $\mu = 1, 2, 3$ .

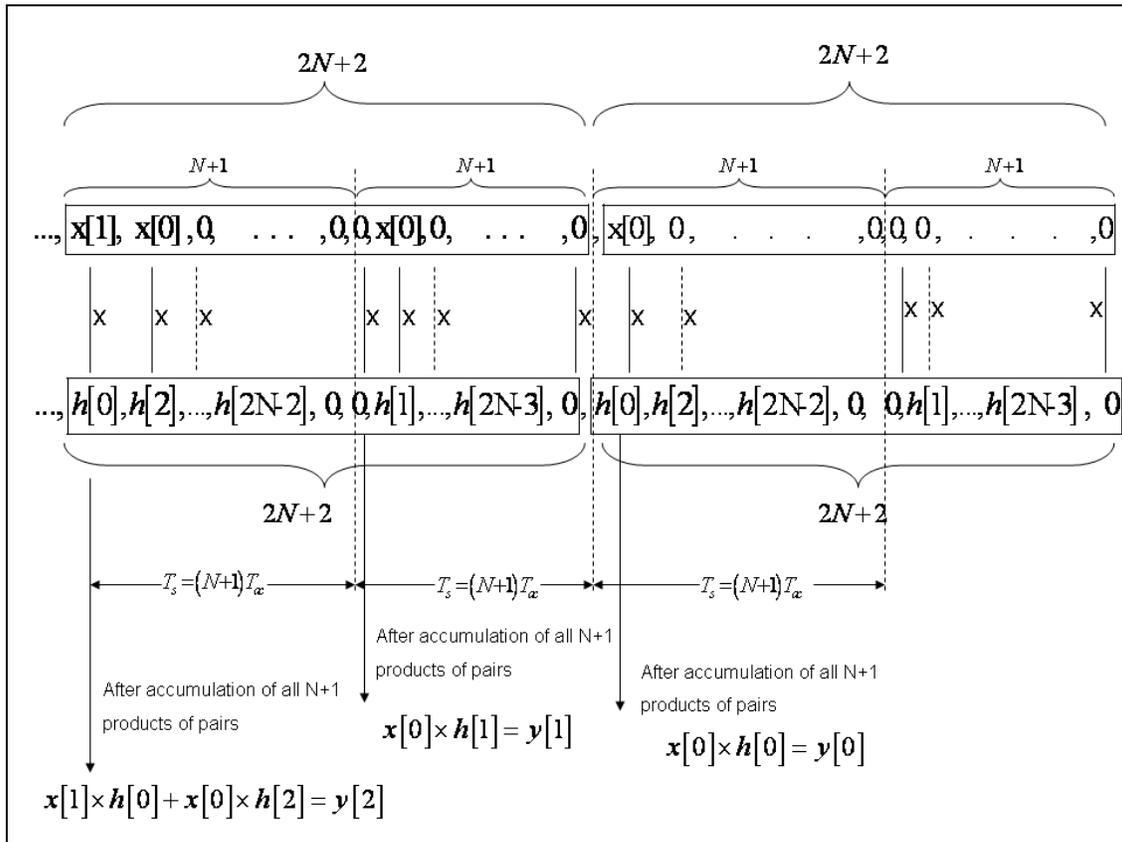


Figure 38. Outcome of DSP48.

Referring to Figure 7, the bitstream outcome  $P$  of the DSP48 can be considered as a set of blocks of length  $2N + 2$  in which the desired coefficients of the interpolation-by-two operation are embedded in every  $(N + 1)^{th}$  and  $(2N + 2)^{th}$  element of each block as it shown in the timing diagram in Figure 36. Therefore by downsampling the data  $P$  by the factor  $N + 1$  (half the factor that was used when the input data was upsampled) the desired interpolation-by-two operation is performed.

*c. Results*

In order to test the performance of the simulation a sinusoidal waveform with frequency  $F = 0.4 \cdot F_s$  (Hz) and sampling frequency  $F_s = 1$  (Hz) is applied as an input.

The FIR filter has been designed as an Equiripple filter and decomposed into two polyphase components with the following characteristics:

Passband: 0-0.4 (in terms of Digital Frequency  $f$  )

Stopband: 0.45-0.5 (in terms of Digital Frequency  $f$  )

Order: 31

The frequency spectrum of the original and the upsampled-by-two signal is shown in Figure 39. We can verify that the frequency spectrum of the upsampled-by-two signal is squeezed (in terms of the digital frequency) by the integer factor of two compared to the frequency spectrum of the original signal. Therefore, the frequency of the original signal is  $f = 0.4$  while the frequency of the upsampled-by-two signal is  $f = 0.4/2 = 0.2$  (where  $f$  is the dimensionless digital frequency).

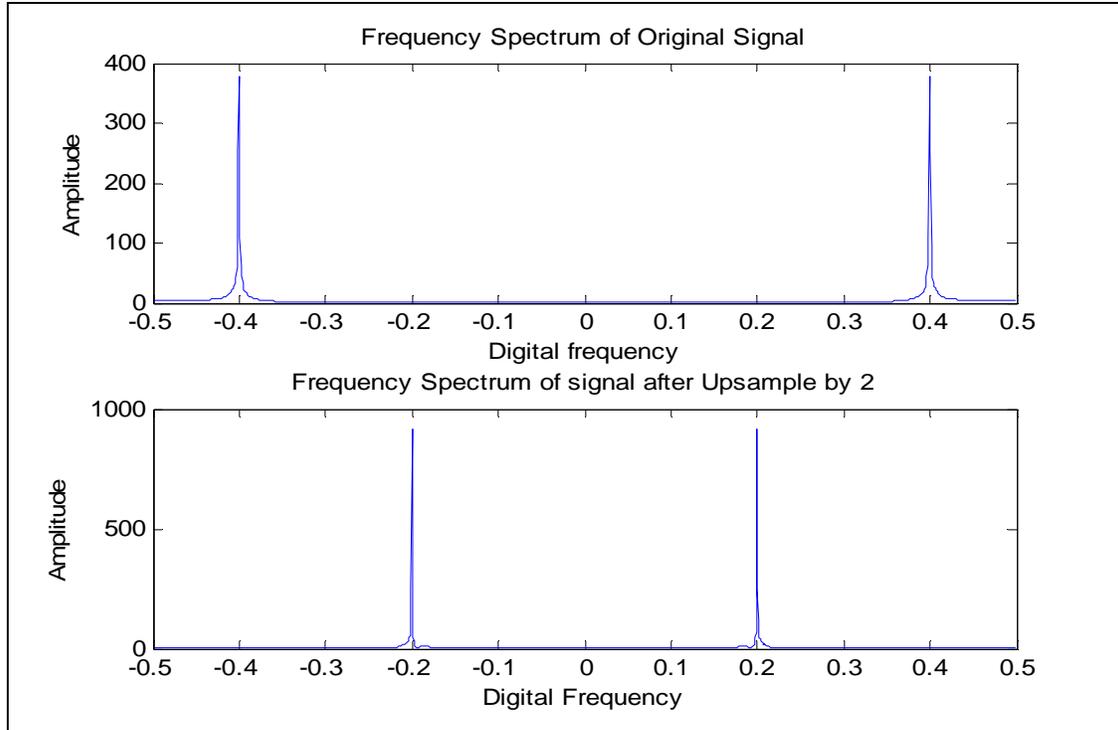


Figure 39. Frequency Spectrum of the Original and Upsampled by Two Signal.

### C. INTERPOLATION BY AN INTEGER FACTOR 'D' WITH FIR MAC AND POLYPHASE DECOMPOSITION

The structure introduced for the interpolation-by-two operation can be easily extended to a more general interpolation-by- $D$  operation for any  $D \geq 2$ . The interpolated signal  $y[n]$  obtained from an input signal  $x[n]$ , which is upsampled by an integer factor  $D$  and then filtered by a FIR filter  $h[n]$  (decomposed into its polyphase components) is given from the following expression:

$$\begin{aligned}
 y[Dn] &= \sum_{k=0}^{N-1} h_0[k]x[n-k], \\
 y[Dn+1] &= \sum_{k=0}^{N-1} h_1[k]x[n-k], \\
 &\dots \\
 y[Dn+D-1] &= \sum_{k=0}^{N-1} h_{D-1}[k]x[n-k],
 \end{aligned} \tag{4.3}$$

with  $D$  integer and  $x[n] = x(nT_s)$ ,  $y[n] = y(nT_s/D)$  the input and output sequences sampled at rates  $F_s = 1/T_s$  and  $DF_s = D/T_s$  respectively.

The  $D$  polyphase components of the FIR filter are defined as:

$$h_\ell[k] = h[kD + \ell] \quad (4.4)$$

with  $\ell = 0, \dots, D-1$  and  $k = 0, \dots, N-1$ .

The simulation needed to perform the interpolation-by- $D$  operation is similar to the interpolation-by-two case. The memory vector for the input data points in the Dual Port Ram has length  $DN$  and it is updated as  $X[(n)_{DN}] \leftarrow x[nD]$ . The data address counter is defined as  $(n)_{DN} = 0, \dots, DN - D - 1$  and it is repeated  $D$  times during the input interval  $T_s$ .

The FIR filter coefficients vector, which is stored in the second memory of the Dual Port Ram, is made of the polyphase components derived from expression (4.4) with total length  $D(N+1)$ .

The input signal is sampled at a rate  $F_s$ , while System Generator (Sysgen) works at a higher sampling rate equal to  $D(N+1)F_s$ . Therefore the input data is upsampled by the integer factor of  $D(N+1)$  with the corresponding Xilinx block

The time representation of ‘data address’ and ‘coefficients address’ sequences are shown in Figure 40. In particular, at time  $(Dn-D)T_s$  the accumulator is initialized as  $a((Dn-D)T_s) = 0$  (where ‘ $a$ ’ denotes the accumulation function). At every subsequent clock cycle  $T_{ac} = \frac{1}{F_{ac}}$  the accumulation will be updated by

$$\begin{aligned}
a((Dn-D)T_s + \lambda T_{ac}) &= a((Dn-D)T_s + (\lambda-1)T_{ac}) + (data\_addr(\lambda) \cdot coeff\_addr(\lambda)), \\
&\dots \\
a((Dn-\ell)T_s + \lambda T_{ac}) &= a((Dn-\ell)T_s + (\lambda-1)T_{ac}) + (data\_addr(\lambda) \cdot coeff\_addr(\lambda)), \\
&\dots \\
a((Dn-1)T_s + \lambda T_{ac}) &= a((Dn-1)T_s + (\lambda-1)T_{ac}) + (data\_addr(\lambda) \cdot coeff\_addr(\lambda)),
\end{aligned}$$

where  $\lambda = 1, \dots, N$ . The outputs  $y[Dn-D], \dots, y[Dn]$  are shown in the timing diagram of Figure 40.

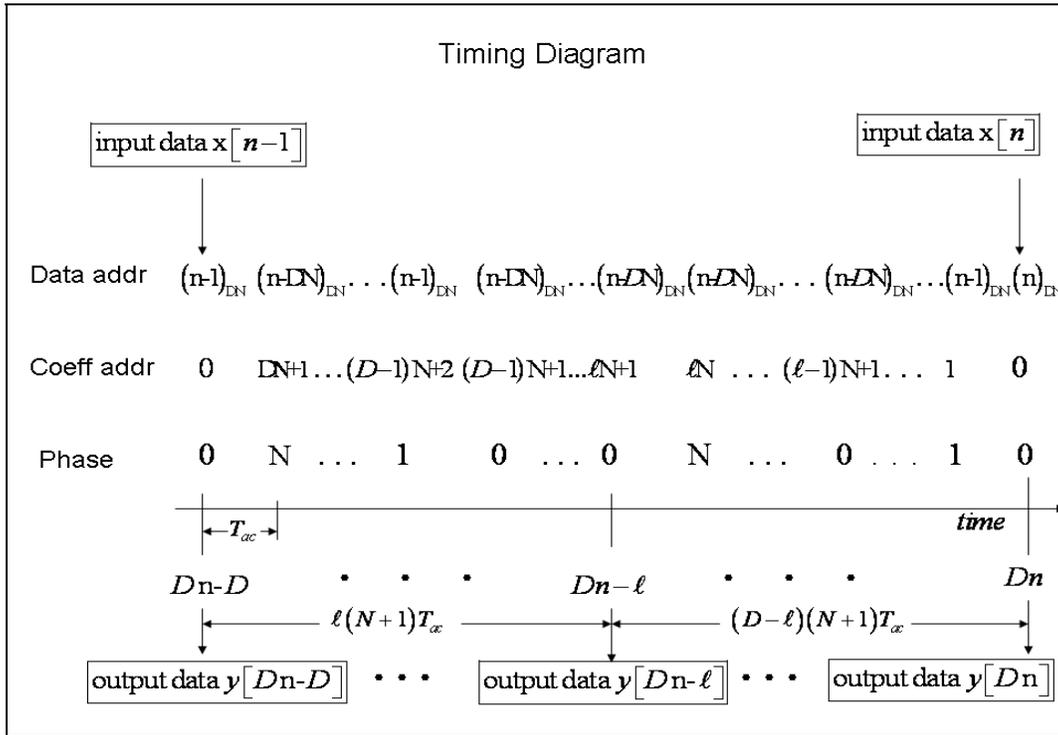


Figure 40. Timing Diagram for Interpolation by D.

Apart from the new vectors that are stored in the Dual Port Ram, this design can be obtained by simple extension of the interpolation-by-two case to the more general interpolation-by-D.

Referring to Figure 7, the bitstream outcome  $P$  of the DSP48 can be considered as a set of blocks of length  $N+1$  in which the desired coefficients of the interpolation-by-D operation are embedded in every  $(N+1)^{th}$  element of each block as it shown in the

timing diagram in Figure 40. Therefore by downsampling the data  $P$  by the factor of  $N+1$  ( $D$  times less the factor which was used when the input data was upsampled) the desired interpolation-by- $D$  operation is performed.

In order to test the performance of the simulation for interpolation factor,  $D = 4$  a sinusoidal waveform with frequency  $F = 0.4 \cdot F_s$  (Hz) and sampling frequency  $F_s = 1$  (Hz) is applied as an input.

The FIR filter has been designed as an Equiripple filter and decomposed into four polyphase components with the following characteristics:

Passband: 0-0.2 (in terms of Digital Frequency  $f$ )

Stopband: 0.25-0.5 (in terms of Digital Frequency  $f$ )

Order: 120

The frequency spectrum of the original and the upsampled by  $D = 4$  signal is shown in Figure 41. We can verify that the frequency spectrum of the upsampled by  $D = 4$  signal is squeezed (in terms of the digital frequency) by the integer factor of four compared to the frequency spectrum of the original signal. Therefore, the frequency of the original signal is  $f = 0.4$ , while the frequency of the upsampled-by-four signal is  $f = 0.4/4 = 0.1$  (where  $f$  is the dimensionless digital frequency).

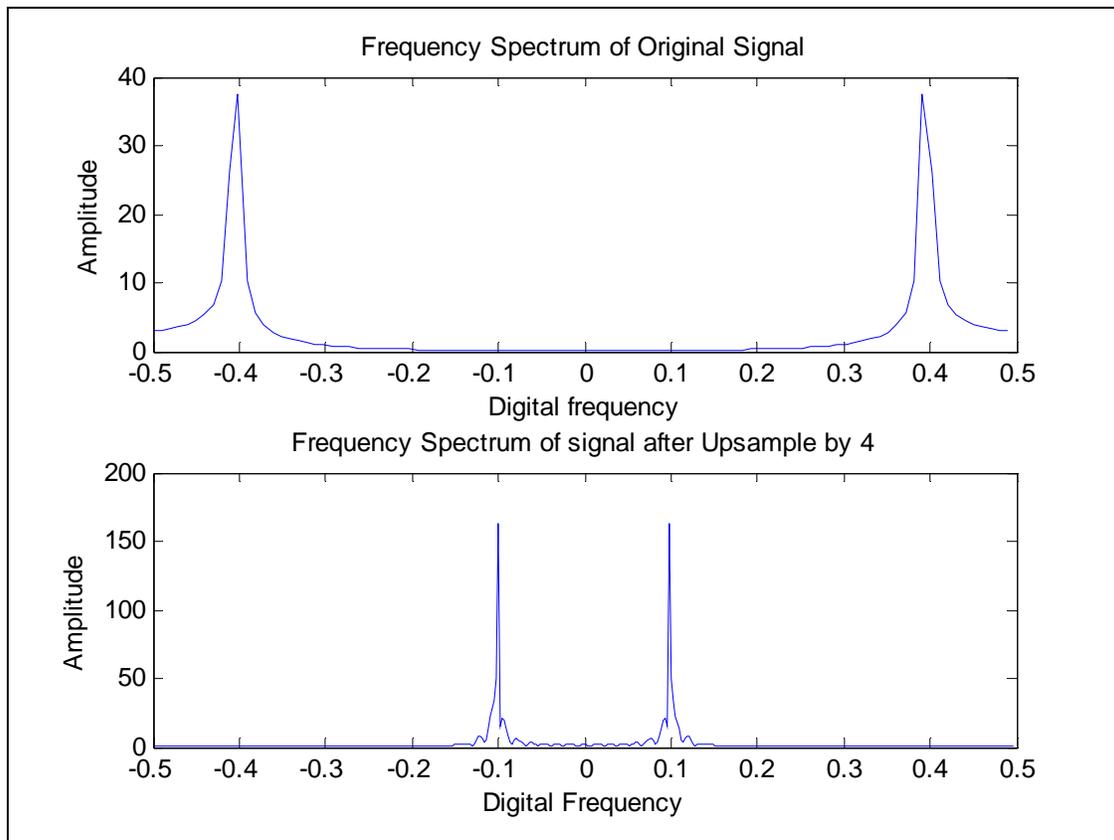


Figure 41. Frequency Spectrum of the Original and Upsampled by  $D = 4$  Signal.

THIS PAGE INTENTIONALLY LEFT BLANK

## V. CONCLUSIONS

### A. SUMMARY OF THE WORK

In this research, we presented an architecture for implementing resampling operations in FPGAs. The particularly interesting feature of this approach is the use of a specific functional block (the DSP48), which is optimized for DSP applications in real time. Although a number of applications are possible, this approach is particularly attractive in the implementation of Software Defined Radios (SDR).

Three classes of DSP operations have been implemented software in the Simulink design environment:

- 1) Finite Impulse Response filter
- 2) Decimation by an integer factor
- 3) Interpolation by an integer factor

All subsystems of the design are fully accessible by the designer at every stage.

The key ingredient was the use of a Multiplier and Accumulator (MAC) architecture carried out from the DSP48 slice, which is an efficient Xilinx block (from the Simulink library) for many DSP applications. The dual input fed to the DSP48 was provided from the Dual Port Ram Xilinx block, which is a memory device that allows the user to specify the width and the values for each memory part in order to perform the three above mentioned operations.

The Xilinx System Generator was used to realize the software performance to a Virtex-4 FPGA, increasing the computation data rate according to each case.

MATLAB code was used to generate the FIR filter and its polyphase decomposition in the design and also to verify the performance providing the desired results in terms of plots demonstrating the corresponding theoretical perspective.

## **B. SUGGESTION FOR FUTURE WORK**

The designs presented in this thesis will be part of a general Software Defined Radio (SDR) implementation. In particular it will be interlaced with both modulation and demodulation processes, so that the whole radio will be implemented in software.

There are several issues to be addressed. The most important is whether this approach can be implemented in real time using a reasonable amount of chip “real estate”. In order to address this problem, higher-level language code needs to be used to implement the algorithm on the chip.

This is part of an ongoing research project.

## LIST OF REFERENCES

- [1] Jeffrey H. Reed, *Software Radio: A modern approach to Radio Engineering*, Prentice Hall, 2002.
- [2] Ruder Bošković Institute,  
[http://www.irb.hr/en/cir/education/courses/fpga/FPGA/fpga\\_sklopovi/](http://www.irb.hr/en/cir/education/courses/fpga/FPGA/fpga_sklopovi/) (Accessed August 12, 2008).
- [3] Xilinx online documentation, System Generator for DSP, User Guide, Release 10.1.1, April 2008  
[http://www.xilinx.com/support/documentation/sw\\_manuals/sysgen\\_user.pdf](http://www.xilinx.com/support/documentation/sw_manuals/sysgen_user.pdf)  
(Accessed August 12, 2008).
- [4] ARM webpage (photo of FPGA Virtex-4 embedded in processing board)  
<http://www.arm.com/rximages/15515.jpg> (Accessed November 7, 2008).
- [5] Xilinx online documentation, System Generator for DSP, Reference Guide, Release 10.1.1, April 2008,  
[http://www.xilinx.com/support/documentation/sw\\_manuals/sysgen\\_ref.pdf](http://www.xilinx.com/support/documentation/sw_manuals/sysgen_ref.pdf)  
(Accessed August 17, 2008).
- [6] Xilinx online documentation, System Generator for DSP, Getting Started Guide, Release 10.1.1, April 2008,  
[http://www.xilinx.com/support/documentation/sw\\_manuals/sysgen\\_gs.pdf](http://www.xilinx.com/support/documentation/sw_manuals/sysgen_gs.pdf),  
(Accessed August 20, 2008).
- [7] Robert D. Strum, Donald E. Kirk, *First Principles of Discrete Systems and Digital Signal Processing*, 1988.
- [8] Roberto Cristi, *Modern Digital Signal Processing*. Thomson Brooks/Cole, 2004.
- [9] P. P. Vaidyanathan, *Multirate Systems and Filter Banks*, Prentice Hall, 1993.
- [10] Xilinx online documentation, Xtreme DSP for Virtex-4 FPGAs, User Guide, UG073(v2.7) May 15,2008  
[http://www.xilinx.com/support/documentation/user\\_guides/ug073.pdf](http://www.xilinx.com/support/documentation/user_guides/ug073.pdf) (Accessed July 18, 2008)
- [11] Ondřej Pirochta, “Hardware Implementations of Digital Filters in FPGA”, Brno University of Technology  
<http://www.urel.feec.vutbr.cz/ra2008/archive/ra2005/papers/401.pdf> (Accessed November 20, 2008)

THIS PAGE INTENTIONALLY LEFT BLANK

## INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center  
Ft. Belvoir, Virginia
2. Dudley Knox Library  
Naval Postgraduate School  
Monterey, California
3. Professor Roberto Cristi  
Naval Postgraduate School  
Monterey, California
4. Associate Professor Craig W. Rasmussen  
Naval Postgraduate School  
Monterey, California
5. Embassy of Greece  
Office of Naval Attaché  
Washington DC 20008